:

Project-2 : COMP1630

:

Table of Contents

Part B – SQL Statements ____

Question 1

____

Question 2

____

Question 3

____

Question 4

____

Question 5

____

Question 6

____

Question 7

____

Question 8

____

Question 9

____

Question 10

____

Part C – INSERT, UPDATE, DELETE and VIEWS Statements ________________________

Question 1

____

Question 2

____

Question 3

____

Question 4

____

Question 5

____

Question 6

____

Question 7

Question 8

Question 9

Question 10

Part D – Stored Procedures and Triggers

Question 1

Question 2

Question 3

Question 4

Question 5

Question 6

Question 7

Question 8

Question 9

3. SUMMARY

4. CHALLENGES

5. SQL SCRIPT

1. INTRODUCTION

This Project has been completed using Microsoft SQL Management Studio to create and query a new database and solve questions. There are 4 main parts of the project: Part A with 6 steps to create a new database and tables, Part B with 10 questions based on SQL statements, Part C with 10 questions regarding INSERT, UPDATE and VIEWS statements and final part D with 9 questions regarding Stored Procedures and Triggers. All the questions are followed by sql statements and evidence of the result using screenshots of all the results. After all the part there is a brief summary and challenges that I face during the project, followed by the copy of the script.

2. SOLUTIONS

Part A – Database and Tables

1.Create a database called Cus_Orders.

```
USE master
GO
```

```
CREATE DATABASE Cus_Orders
GO
USE Cus_orders
GO
```

2. Create a user defined data types for all similar Primary Key attribute columns (e.g. order_id, product_id, title_id), to ensure the same data type, length and null ability. See Pages 12/13 for specification

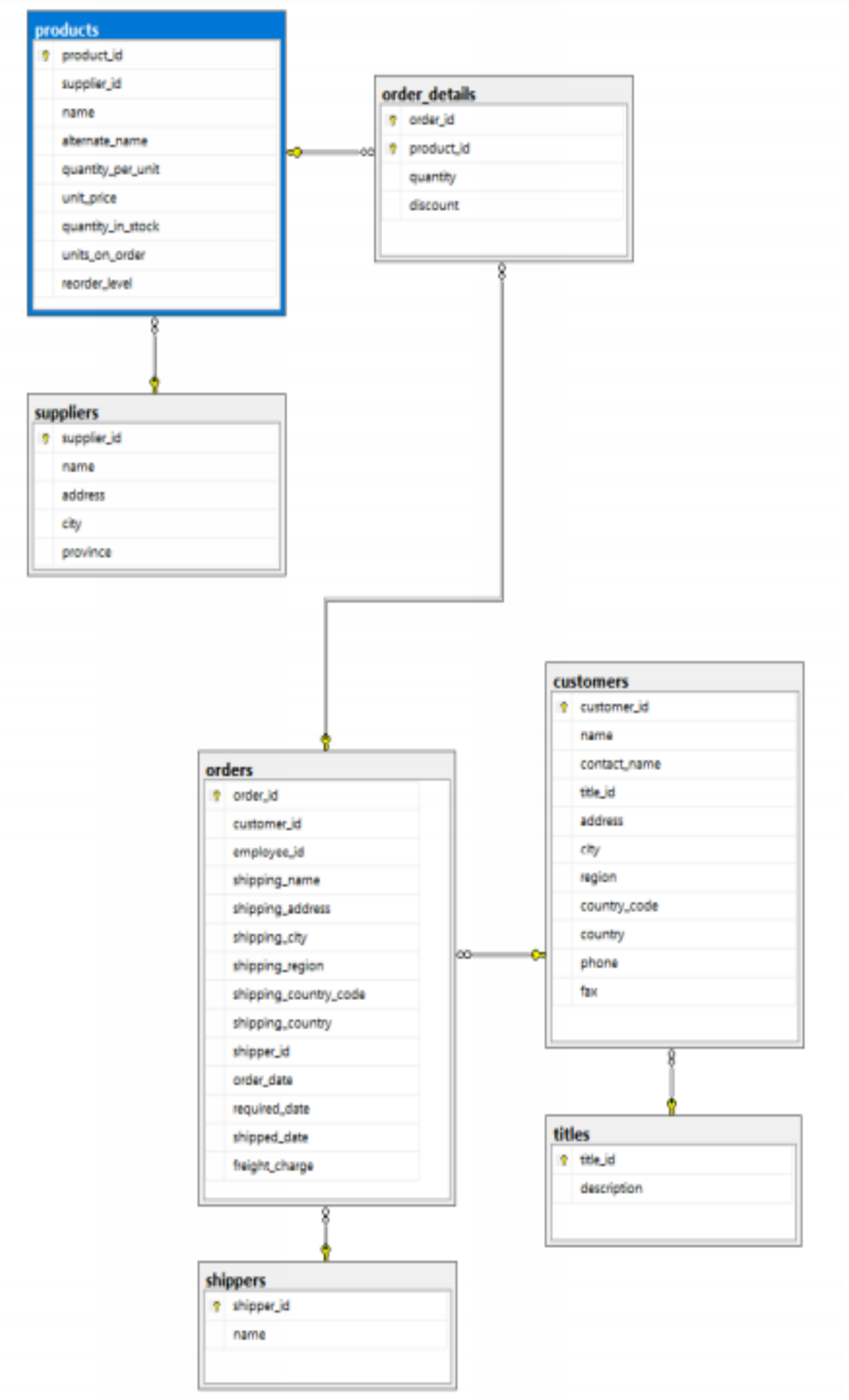
```
CREATE TYPE cusid FROM char(5) NOT NULL;
CREATE TYPE intid FROM int NOT NULL;
GO
```

3. Create the following tables (see column information on pages 12 and 13):

   customers
   orders
   order_details
   products
   shippers
   suppliers
   titles

```
CREATE TABLE customers (
customer_id cusid,
name varchar (50) NOT NULL,
contact_name varchar (30),
title_id char(3) NOT NULL,
address varchar (50),
city varchar(20),
region varchar(15),
country_code varchar(10),
country varchar(15),
phone varchar(20),
fax varchar(20)
);
CREATE TABLE orders (
order_id intid,
customer_id cusid,
employee_id int NOT NULL,
shipping_name varchar(50),
shipping_address varchar(50),
shipping_city varchar(20),
shipping_region varchar(15),
shipping_country_code varchar(10),
shipping_country varchar(15),
shipper_id int NOT NULL,
order_date datetime,
required_date datetime,
shipped_date datetime,
freight_charge money
);
CREATE TABLE order_details (
order_id intid,
product_id intid,
quantity int NOT NULL,
discount float NOT NULL
);
CREATE TABLE products (
product_id intid,
supplier_id int NOT NULL,
name varchar(40) NOT NULL,
alternate_name varchar(40),
quantity_per_unit varchar(25),
unit_price money,
quantity_in_stock int,
units_on_order int,
reorder_level int
);
CREATE TABLE shippers (
shipper_id int IDENTITY NOT NULL,

name varchar(20)
);
CREATE TABLE suppliers (
supplier_id int IDENTITY NOT NULL,
name varchar(40),
address varchar(30),
city varchar(20),
province char(2)
);
CREATE TABLE titles (
title_id char(3) NOT NULL,
description varchar(35)
);
GO
```

The database diagram is shown on the preceding page



4.Set the primary keys and foreign keys for the tables.

```
ALTER TABLE customers
ADD PRIMARY KEY (customer_id);

ALTER TABLE shippers
ADD PRIMARY KEY (shipper_id);
```

```
ALTER TABLE titles
ADD PRIMARY KEY (title_id);

ALTER TABLE orders
ADD PRIMARY KEY (order_id);

ALTER TABLE suppliers
ADD PRIMARY KEY (supplier_id);

ALTER TABLE products
ADD PRIMARY KEY (product_id);

ALTER TABLE order_details
ADD PRIMARY KEY (order_id, product_id);
GO

ALTER TABLE customers
ADD CONSTRAINT fk_cust_titles FOREIGN KEY (title_id)
REFERENCES titles(title_id);

ALTER TABLE orders
ADD CONSTRAINT fk_orders_cust FOREIGN KEY (customer_id)
REFERENCES customers(customer_id);

ALTER TABLE orders
ADD CONSTRAINT fk_orders_shippers FOREIGN KEY (shipper_id)
REFERENCES shippers(shipper_id);

ALTER TABLE order_details
ADD CONSTRAINT fk_order_details_orders FOREIGN KEY (order_id)
REFERENCES orders(order_id);

ALTER TABLE order_details
ADD CONSTRAINT fk_order_details_products FOREIGN KEY (product_id)
REFERENCES products(product_id);

ALTER TABLE products
ADD CONSTRAINT fk_products_suppliers FOREIGN KEY (supplier_id)
REFERENCES suppliers(supplier_id);
GO
```

5.Set the constraints as follows: -
customers tables - country should default to Canada
orders table - required_date should default to today's date plus ten days
order details table - quantity must be greater than or equal to 1
products table - reorder_level must be greater than or equal to 1
- quantity_in_stock value must not be greater than 150
suppliers table - province should default to B

```
ALTER TABLE customers
ADD CONSTRAINT default_country
DEFAULT('Canada') FOR country;

ALTER TABLE orders
ADD CONSTRAINT default_required_date
DEFAULT(GETDATE() + 10) FOR required_date;

ALTER TABLE order_details
ADD CONSTRAINT min_quant
```

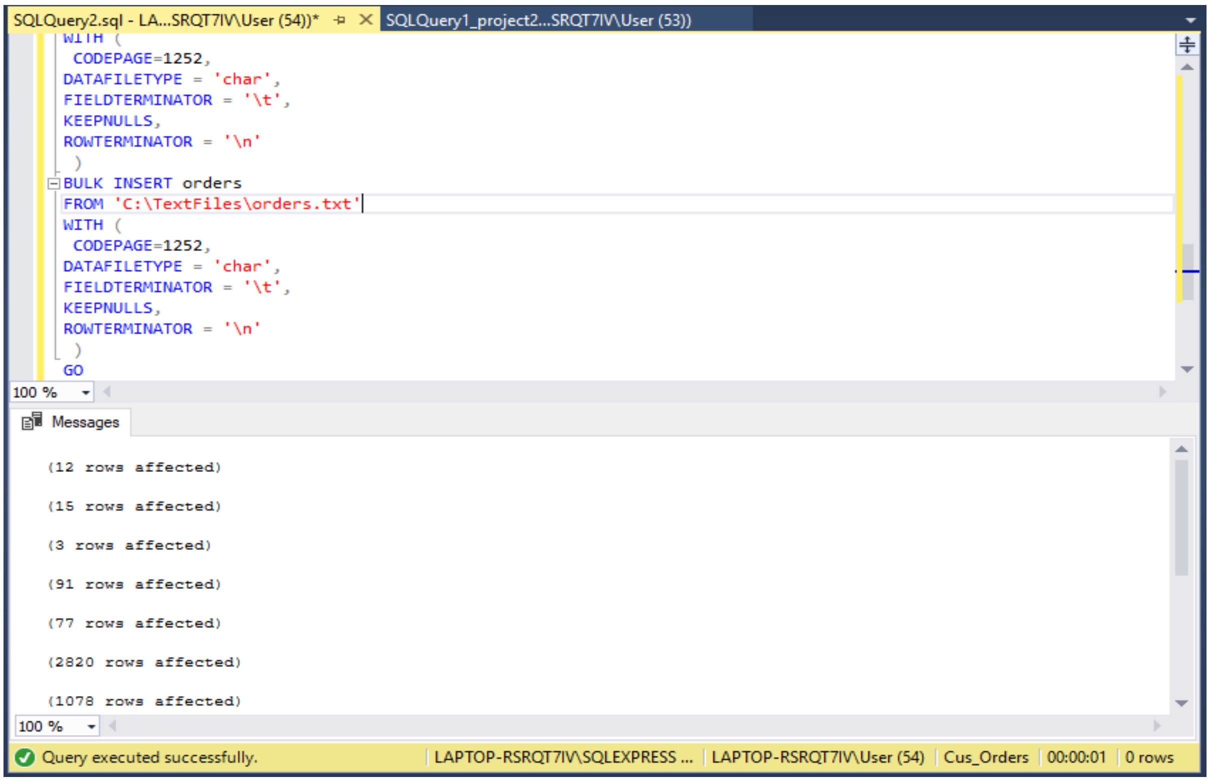
```
 CHECK (quantity >= 1);

ALTER TABLE products
ADD CONSTRAINT min_reorder_level
 CHECK (reorder_level >= 1);

ALTER TABLE products
ADD CONSTRAINT max_quant_in_stock
 CHECK (quantity_in_stock < 150);

ALTER TABLE suppliers
ADD CONSTRAINT default_province
 DEFAULT('BC') FOR province;
 GO
```

6.Load the data into your created tables using the following files:

customers.txt	into the customers table	(91 rows)
orders.txt	into the orders table	(1078 rows)
order_details.txt	into the order_details table	(2820 rows)
products.txt	into the products table	(77 rows)
shippers.txt	into the shippers table	(3 rows)
suppliers.txt	into the suppliers table	(15 rows)
titles.txt	into the titles table	(12 rows)
employees.txt	into the employees table which is created in Part C (See	

```
BULK INSERT titles
FROM 'C:\TextFiles\titles.txt'
WITH (
               CODEPAGE=1252,
             DATAFILETYPE = 'char',
             FIELDTERMINATOR = '\t',
             KEEPNULLS,
             ROWTERMINATOR = '\n'
        )

BULK INSERT suppliers
FROM 'C:\TextFiles\suppliers.txt'
WITH (
               CODEPAGE=1252,
             DATAFILETYPE = 'char',
             FIELDTERMINATOR = '\t',
             KEEPNULLS,
             ROWTERMINATOR = '\n'
         )


BULK INSERT shippers
FROM 'C:\TextFiles\shippers.txt'
WITH (
               CODEPAGE=1252,
             DATAFILETYPE = 'char',
             FIELDTERMINATOR = '\t',
             KEEPNULLS,
             ROWTERMINATOR = '\n'
         )
```

```
BULK INSERT customers
FROM 'C:\TextFiles\customers.txt'
WITH (
                CODEPAGE=1252,
             DATAFILETYPE = 'char',
             FIELDTERMINATOR = '\t',
             KEEPNULLS,
             ROWTERMINATOR = '\n'
         )

BULK INSERT products
FROM 'C:\TextFiles\products.txt'
WITH (
                CODEPAGE=1252,
             DATAFILETYPE = 'char',
             FIELDTERMINATOR = '\t',
             KEEPNULLS,
             ROWTERMINATOR = '\n'
         )

BULK INSERT order_details
FROM 'C:\TextFiles\order_details.txt'
WITH (
                CODEPAGE=1252,
             DATAFILETYPE = 'char',
             FIELDTERMINATOR = '\t',
             KEEPNULLS,
             ROWTERMINATOR = '\n'
         )

BULK INSERT orders
FROM 'C:\TextFiles\orders.txt'
WITH (
                CODEPAGE=1252,
             DATAFILETYPE = 'char',
             FIELDTERMINATOR = '\t',
             KEEPNULLS,
             ROWTERMINATOR = '\n'
         )
GO


BULK INSERT employee
FROM 'C:\TextFiles\employee.txt'
WITH (          CODEPAGE=1252,
             DATAFILETYPE = 'char',
             FIELDTERMINATOR = '\t',
             KEEPNULLS,
             ROWTERMINATOR = '\n'
        )
GO
```


SQLQuery2.sql - LA...SRQT7IV\User (54))* SQLQuery1_project2...SRQT7IV\User (53))

```
WITH (
 CODEPAGE=1252,
DATAFILETYPE = 'char',
FIELDTERMINATOR = '\t',
KEEPNULLS,
ROWTERMINATOR = '\n'
 )
BULK INSERT orders
FROM 'C:\TextFiles\orders.txt'
WITH (
 CODEPAGE=1252,
DATAFILETYPE = 'char',
FIELDTERMINATOR = '\t',
KEEPNULLS,
ROWTERMINATOR = '\n'
 )
GO
```

Messages

```
(12 rows affected)

(15 rows affected)

(3 rows affected)

(91 rows affected)

(77 rows affected)

(2820 rows affected)

(1078 rows affected)
```

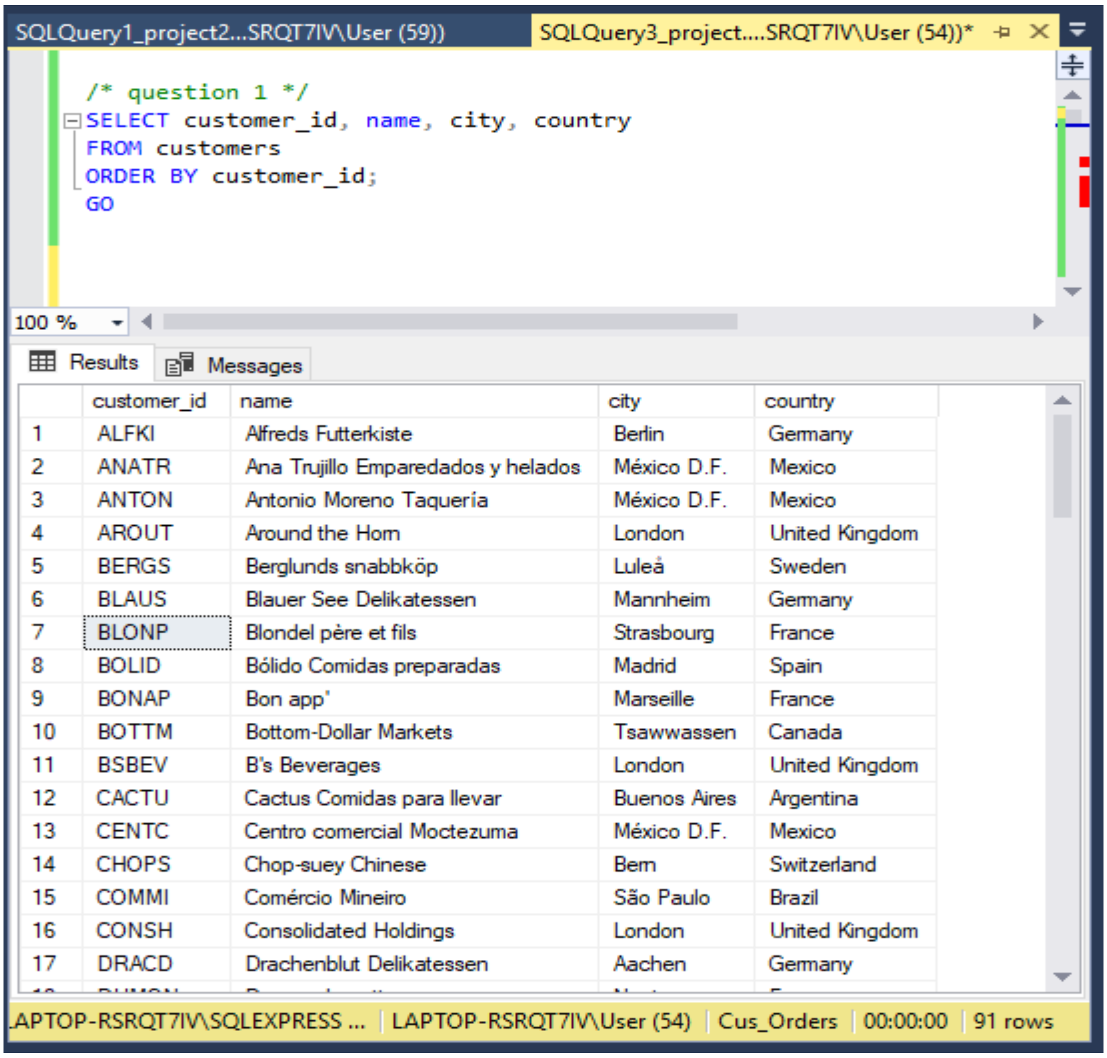
Query executed successfully. | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (54) | Cus_Orders | 00:00:01 | 0 rows

Part B – SQL Statements

1. List the customer id, name, city, and country from the customer table. Order the result set by the **customer id**. The query should produce the result set listed below.

```
customer_id    name                               city           country
-------------- ---------------------------------- -------------  ---------------
ALFKI          Alfreds Futterkiste                Berlin         Germany
ANATR          Ana Trujillo Emparedados y helados México D.F.    Mexico
ANTON          Antonio Moreno Taquería            México D.F.    Mexico
AROUT          Around the Horn                    London         United Kingdom
BERGS          Berglunds snabbköp                 Luleå          Sweden
...
WHITC          White Clover Markets               Seattle        United States
WILMK          Wilman Kala                        Helsinki       Finland
WOLZA          Wolski  Zajazd                     Warszawa       Poland

(91 row(s) affected)
```

```
/* question 1 */
SELECT customer_id, name, city, country
FROM customers
ORDER BY customer_id;
GO
```


SQLQuery1_project2...SRQT7IV\User (59))
SQLQuery3_project....SRQT7IV\User (54))*
100 %
Results
Messages

```
/* question 1 */
SELECT customer_id, name, city, country
FROM customers
ORDER BY customer_id;
GO
```

	customer_id	name	city	country
1	ALFKI	Alfreds Futterkiste	Berlin	Germany
2	ANATR	Ana Trujillo Emparedados y helados	México D.F.	Mexico
3	ANTON	Antonio Moreno Taquería	México D.F.	Mexico
4	AROUT	Around the Horn	London	United Kingdom
5	BERGS	Berglunds snabbköp	Luleå	Sweden
6	BLAUS	Blauer See Delikatessen	Mannheim	Germany
7	BLONP	Blondel père et fils	Strasbourg	France
8	BOLID	Bólido Comidas preparadas	Madrid	Spain
9	BONAP	Bon app'	Marseille	France
10	BOTTM	Bottom-Dollar Markets	Tsawwassen	Canada
11	BSBEV	B's Beverages	London	United Kingdom
12	CACTU	Cactus Comidas para llevar	Buenos Aires	Argentina
13	CENTC	Centro comercial Moctezuma	México D.F.	Mexico
14	CHOPS	Chop-suey Chinese	Bern	Switzerland
15	COMMI	Comércio Mineiro	São Paulo	Brazil
16	CONSH	Consolidated Holdings	London	United Kingdom
17	DRACD	Drachenblut Delikatessen	Aachen	Germany

LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (54) | Cus_Orders | 00:00:00 | 91 rows

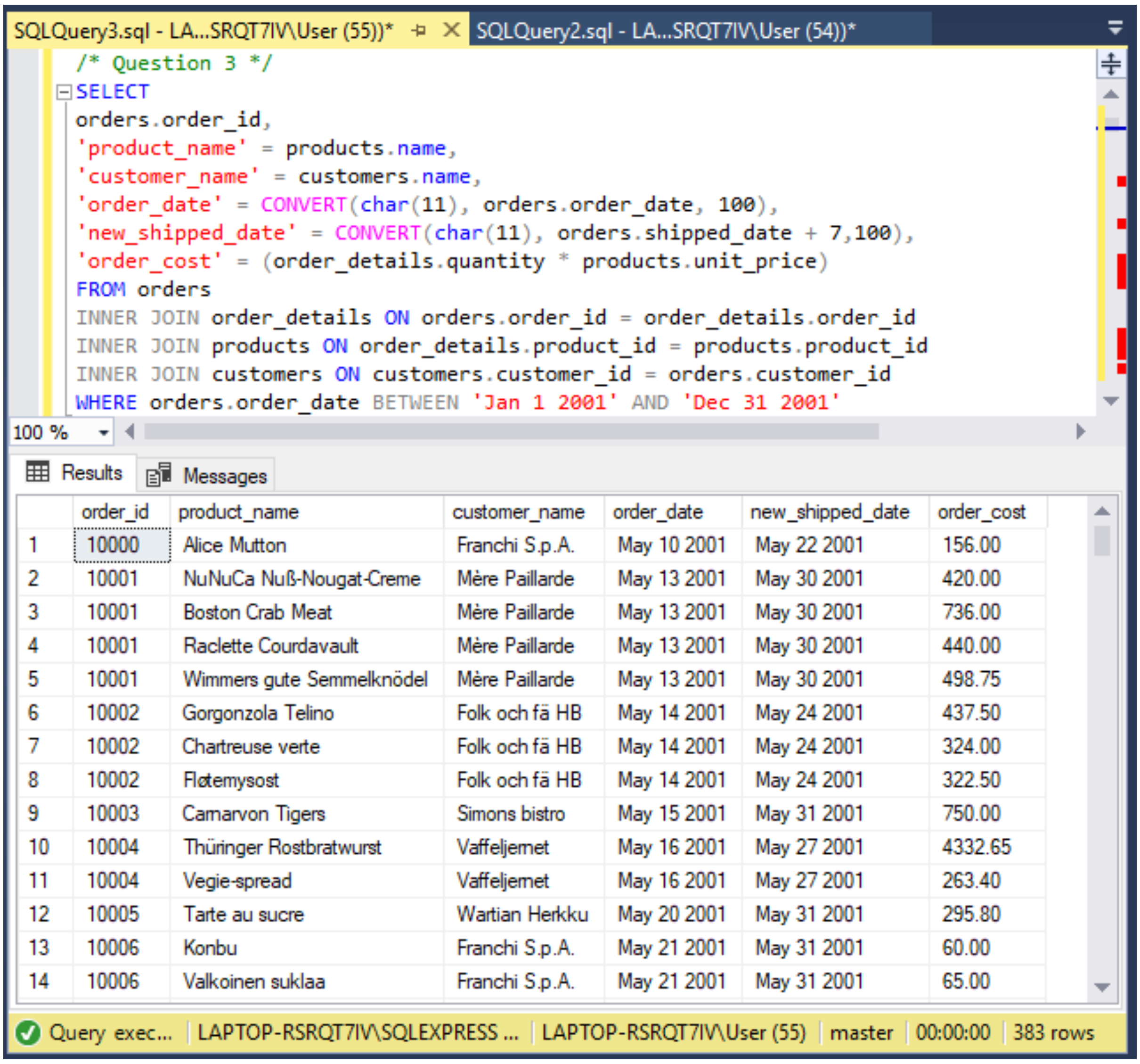
2. Add a new column called active to the customers table using the ALTER statement. The only valid values are 1 or 0. The default should be 1.

```
ALTER TABLE customers
ADD active BIT NOT NULL
CONSTRAINT default_active DEFAULT(1);
GO
```

3. List all the orders where the order date is between **January 1** and **December 31, 2001**. Display the order id, order date, and a new shipped date calculated by adding 7 days to the shipped date from the orders table, the product name from the product table, the customer name from the customer table, and the cost of the order. Format the date order date and the shipped date as **MON DD YYYY**. Use the formula (quantity * unit_price) to calculate the cost of the order. The query should produce the result set listed below.

```
order_id    product_name                  customer_name      order_date     new_shipped_date   order_cost
--------    -----------------------       -------------      -----------    ----------------   ----------
10000       Alice Mutton                  Franchi S.p.A.     May 10 2001    May 22 2001        156.0000
10001       NuNuCa Nuß-Nougat-Crème       Mère Paillarde     May 13 2001    May 30 2001        420.0000
10001       Boston Crab Meat              Mère Paillarde     May 13 2001    May 30 2001        736.0000
10001       Raclette Courdavault          Mère Paillarde     May 13 2001    May 30 2001        440.0000
10001       Wimmers gute Semmelknödel     Mère Paillarde     May 13 2001    May 30 2001        498.7500
...
10138       Inlagd Sill                   Du monde entire    Dec 27 2001    Jan 10 2002        228.0000
10138       Louisiana Hot Spiced Okra     Du monde entire    Dec 27 2001    Jan 10 2002        204.0000
10139       Camembert Pierrot             Vaffeljernet       Dec 30 2001    Jan 16 2002        680.0000

(383 row(s) affected)
```

```sql
/* question 3 */
SELECT
orders.order_id,
'product_name' = products.name,
'customer_name' = customers.name,
'order_date' = CONVERT(char(11), orders.order_date, 100),
'new_shipped_date' = CONVERT(char(11), orders.shipped_date + 7,100),
'order_cost' = (order_details.quantity * products.unit_price)
FROM orders
INNER JOIN order_details ON orders.order_id = order_details.order_id
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN customers ON customers.customer_id = orders.customer_id
WHERE orders.order_date BETWEEN 'Jan 1 2001' AND 'Dec 31 2001'
GO
```


SQLQuery3.sql - LA...SRQT7IV\User (55))*
SQLQuery2.sql - LA...SRQT7IV\User (54))*
100 %
Results
Messages

```
/* Question 3 */
SELECT
orders.order_id,
'product_name' = products.name,
'customer_name' = customers.name,
'order_date' = CONVERT(char(11), orders.order_date, 100),
'new_shipped_date' = CONVERT(char(11), orders.shipped_date + 7,100),
'order_cost' = (order_details.quantity * products.unit_price)
FROM orders
INNER JOIN order_details ON orders.order_id = order_details.order_id
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN customers ON customers.customer_id = orders.customer_id
WHERE orders.order_date BETWEEN 'Jan 1 2001' AND 'Dec 31 2001'
```

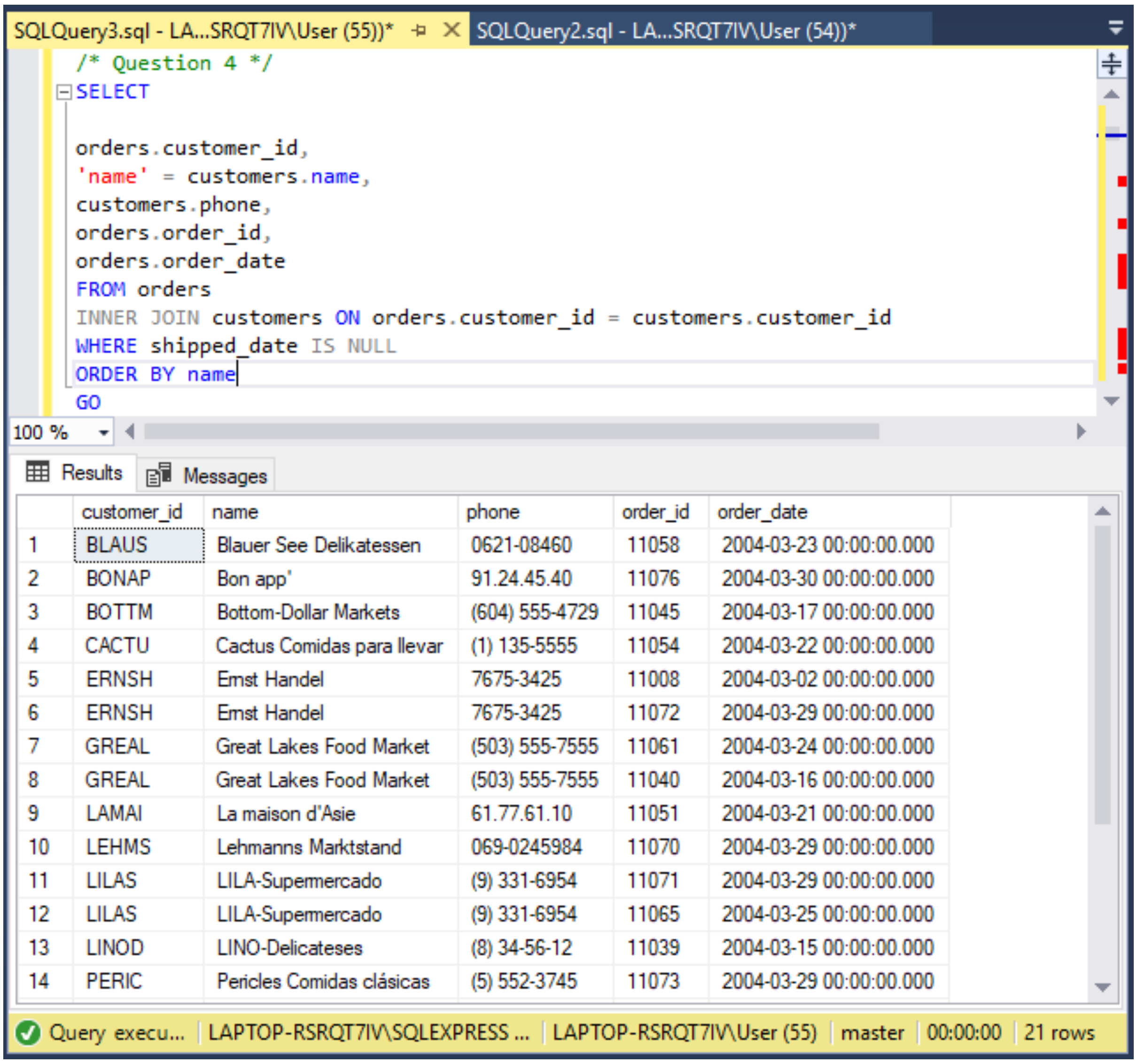
	order_id	product_name	customer_name	order_date	new_shipped_date	order_cost
1	10000	Alice Mutton	Franchi S.p.A.	May 10 2001	May 22 2001	156.00
2	10001	NuNuCa Nuß-Nougat-Creme	Mère Paillarde	May 13 2001	May 30 2001	420.00
3	10001	Boston Crab Meat	Mère Paillarde	May 13 2001	May 30 2001	736.00
4	10001	Raclette Courdavault	Mère Paillarde	May 13 2001	May 30 2001	440.00
5	10001	Wimmers gute Semmelknödel	Mère Paillarde	May 13 2001	May 30 2001	498.75
6	10002	Gorgonzola Telino	Folk och fä HB	May 14 2001	May 24 2001	437.50
7	10002	Chartreuse verte	Folk och fä HB	May 14 2001	May 24 2001	324.00
8	10002	Fløtemysost	Folk och fä HB	May 14 2001	May 24 2001	322.50
9	10003	Carnarvon Tigers	Simons bistro	May 15 2001	May 31 2001	750.00
10	10004	Thüringer Rostbratwurst	Vaffeljernet	May 16 2001	May 27 2001	4332.65
11	10004	Vegie-spread	Vaffeljernet	May 16 2001	May 27 2001	263.40
12	10005	Tarte au sucre	Wartian Herkku	May 20 2001	May 31 2001	295.80
13	10006	Konbu	Franchi S.p.A.	May 21 2001	May 31 2001	60.00
14	10006	Valkoinen suklaa	Franchi S.p.A.	May 21 2001	May 31 2001	65.00

Query exec... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (55) | master | 00:00:00 | 383 rows

4. List all the orders that have **not** been shipped. Display the customer id, name and phone number from the customers table, and the order id and order date from the orders table. Order the result set by the customer name. The query should produce the result set listed below.

```
customer_id  name                        phone          order_id  order_date
-----------  --------------------------  -------------  --------  --------------------
BLAUS        Blauer See Delikatessen     0621-08460     11058     2004-03-23 00:00:00.000
BONAP        Bon app'                     91.24.45.40   11076     2004-03-30 00:00:00.000
ERNSH        Ernst Handel                7675-3425      11008     2004-03-02 00:00:00.000
.....
RICAR        Ricardo Adocicados          (21) 555-3412  11059     2004-03-23 00:00:00.000
RICSU        Richter Supermarkt          0897-034214    11075     2004-03-30 00:00:00.000
SIMOB         Simons bistro              31 12 34 56    11074     2004-03-30 00:00:00.000

(21 row(s) affected)
```

```
/* Question 4 */
SELECT

orders.customer_id,
'name' = customers.name,
customers.phone,
orders.order_id,
orders.order_date
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
WHERE shipped_date IS NULL
ORDER BY name
GO
```



SQLQuery3.sql - LA...SRQT7IV\User (55))*   SQLQuery2.sql - LA...SRQT7IV\User (54))*

```
/* Question 4 */
SELECT

orders.customer_id,
'name' = customers.name,
customers.phone,
orders.order_id,
orders.order_date
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
WHERE shipped_date IS NULL
ORDER BY name
GO
```

100 %

Results | Messages

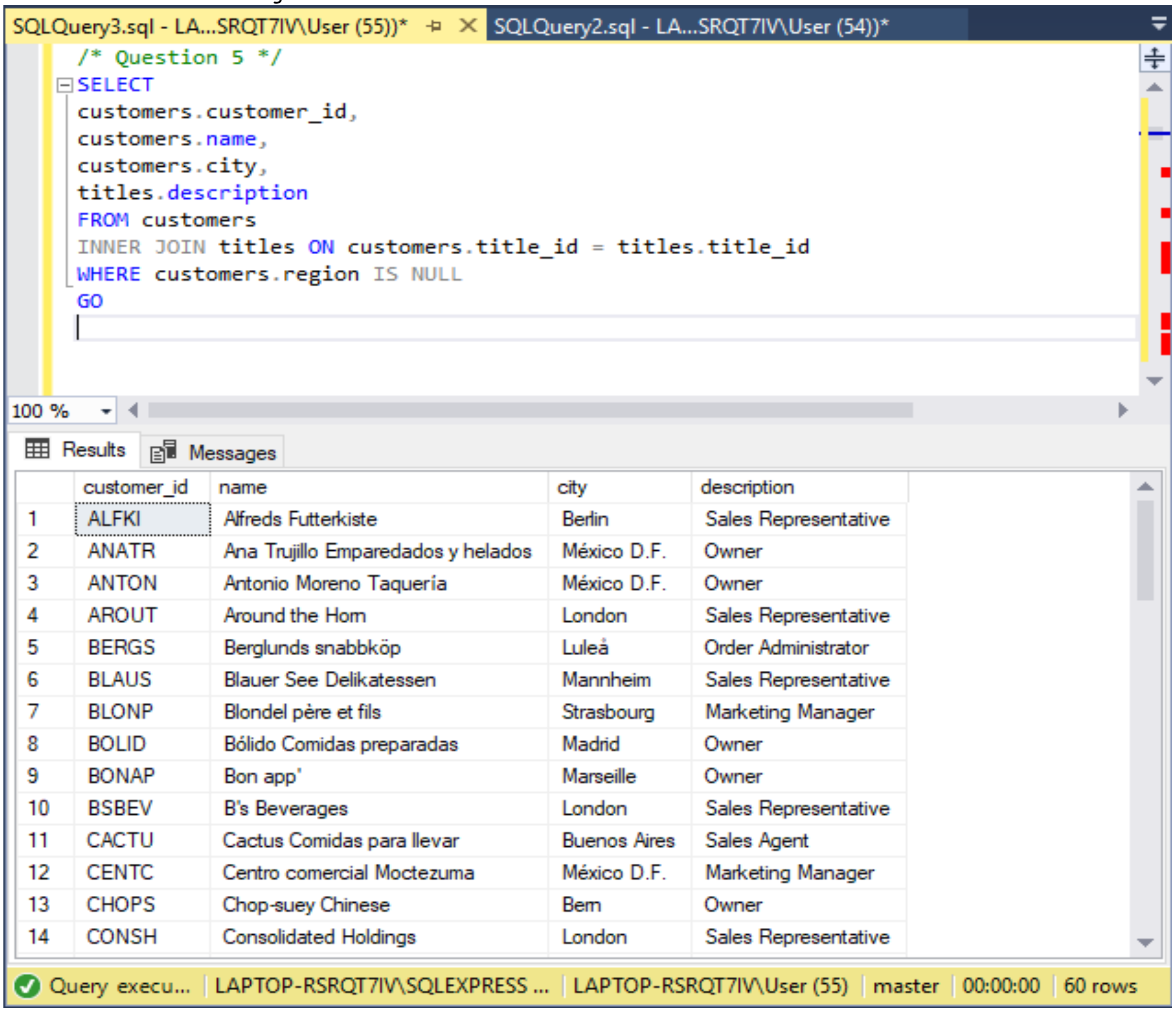
	customer_id	name	phone	order_id	order_date
1	BLAUS	Blauer See Delikatessen	0621-08460	11058	2004-03-23 00:00:00.000
2	BONAP	Bon app'	91.24.45.40	11076	2004-03-30 00:00:00.000
3	BOTTM	Bottom-Dollar Markets	(604) 555-4729	11045	2004-03-17 00:00:00.000
4	CACTU	Cactus Comidas para llevar	(1) 135-5555	11054	2004-03-22 00:00:00.000
5	ERNSH	Ernst Handel	7675-3425	11008	2004-03-02 00:00:00.000
6	ERNSH	Ernst Handel	7675-3425	11072	2004-03-29 00:00:00.000
7	GREAL	Great Lakes Food Market	(503) 555-7555	11061	2004-03-24 00:00:00.000
8	GREAL	Great Lakes Food Market	(503) 555-7555	11040	2004-03-16 00:00:00.000
9	LAMAI	La maison d'Asie	61.77.61.10	11051	2004-03-21 00:00:00.000
10	LEHMS	Lehmanns Marktstand	069-0245984	11070	2004-03-29 00:00:00.000
11	LILAS	LILA-Supermercado	(9) 331-6954	11071	2004-03-29 00:00:00.000
12	LILAS	LILA-Supermercado	(9) 331-6954	11065	2004-03-25 00:00:00.000
13	LINOD	LINO-Delicateses	(8) 34-56-12	11039	2004-03-15 00:00:00.000
14	PERIC	Pericles Comidas clásicas	(5) 552-3745	11073	2004-03-29 00:00:00.000

Query execu... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (55) | master | 00:00:00 | 21 rows

5. List all the customers where the region is **NULL**. Display the customer id, name, and city from the customers table, and the title description from the titles table. The query should produce the result set listed below.

```
customer_id   name                               city            description
-----------   ---------------------------------  --------------  ---------------------
ALFKI         Alfreds Futterkiste                Berlin          Sales Representative
ANATR         Ana Trujillo Emparedados y helados México D.F.     Owner
ANTON         Antonio Moreno Taquería            México D.F.     Owner
AROUT         Around the Horn                    London          Sales Representative
BERGS         Berglunds snabbköp                 Luleå           Order Administrator
...
WARTH         Wartian Herkku                     Oulu            Accounting Manager
WILMK         Wilman Kala                        Helsinki        Owner/Marketing Assistant
WOLZA         Wolski  Zajazd                     Warszawa        Owner

(60 row(s) affected)
```

```
/* Question 5 */
SELECT
customers.customer_id,
customers.name,
customers.city,
titles.description
FROM customers
INNER JOIN titles ON customers.title_id = titles.title_id
WHERE customers.region IS NULLGO
```


SQLQuery3.sql - LA...SRQT7IV\User (55))*
SQLQuery2.sql - LA...SRQT7IV\User (54))*
100 %
Results
Messages

```
/* Question 5 */
SELECT
customers.customer_id,
customers.name,
customers.city,
titles.description
FROM customers
INNER JOIN titles ON customers.title_id = titles.title_id
WHERE customers.region IS NULL
GO
```

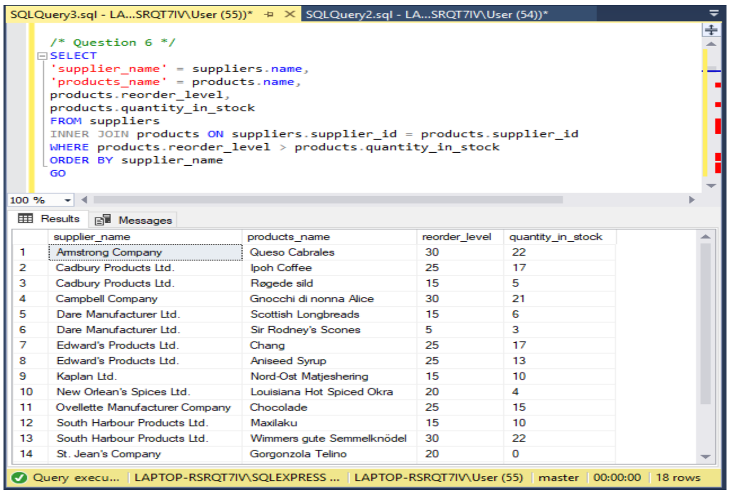
	customer_id	name	city	description
1	ALFKI	Alfreds Futterkiste	Berlin	Sales Representative
2	ANATR	Ana Trujillo Emparedados y helados	México D.F.	Owner
3	ANTON	Antonio Moreno Taquería	México D.F.	Owner
4	AROUT	Around the Horn	London	Sales Representative
5	BERGS	Berglunds snabbköp	Luleå	Order Administrator
6	BLAUS	Blauer See Delikatessen	Mannheim	Sales Representative
7	BLONP	Blondel père et fils	Strasbourg	Marketing Manager
8	BOLID	Bólido Comidas preparadas	Madrid	Owner
9	BONAP	Bon app'	Marseille	Owner
10	BSBEV	B's Beverages	London	Sales Representative
11	CACTU	Cactus Comidas para llevar	Buenos Aires	Sales Agent
12	CENTC	Centro comercial Moctezuma	México D.F.	Marketing Manager
13	CHOPS	Chop-suey Chinese	Bern	Owner
14	CONSH	Consolidated Holdings	London	Sales Representative

Query execu... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (55) | master | 00:00:00 | 60 rows

6. List the products where the reorder level is **higher than** the quantity in stock. Display the supplier name from the suppliers table, the product name, reorder level, and quantity in stock from the products table. Order the result set by the supplier name. The query should produce the result set listed below.

```
supplier_name              product_name            reorder_level    quantity_in_stock
------------------------   ---------------------   ---------------  -----------------
Armstrong Company          Queso Cabrales          30               22
Cadbury Products Ltd.      Ipoh Coffee             25               17
Cadbury Products Ltd.      Røgede sild             15               5
Campbell Company           Gnocchi di nonna Alice  30               21
Dare Manufacturer Ltd.     Scottish Longbreads     15               6
...
Steveston Export Company   Gravad lax              25               11
Steveston Export Company   Outback Lager           30               15
Yves Delorme Ltd.          Longlife Tofu           5                4

(18 row(s) affected)
```

```sql
/* Question 6 */
SELECT
'supplier_name' = suppliers.name,
'products_name' = products.name,
products.reorder_level,
products.quantity_in_stock
FROM suppliers
INNER JOIN products ON suppliers.supplier_id = products.supplier_id
WHERE products.reorder_level > products.quantity_in_stock
ORDER BY supplier_name
GO
```


SQLQuery3.sql - LA...SRQT7IV\User (55))*
SQLQuery2.sql - LA...SRQT7IV\User (54))*
100 %
Results
Messages

```sql
/* Question 6 */
SELECT
'supplier_name' = suppliers.name,
'products_name' = products.name,
products.reorder_level,
products.quantity_in_stock
FROM suppliers
INNER JOIN products ON suppliers.supplier_id = products.supplier_id
WHERE products.reorder_level > products.quantity_in_stock
ORDER BY supplier_name
GO
```

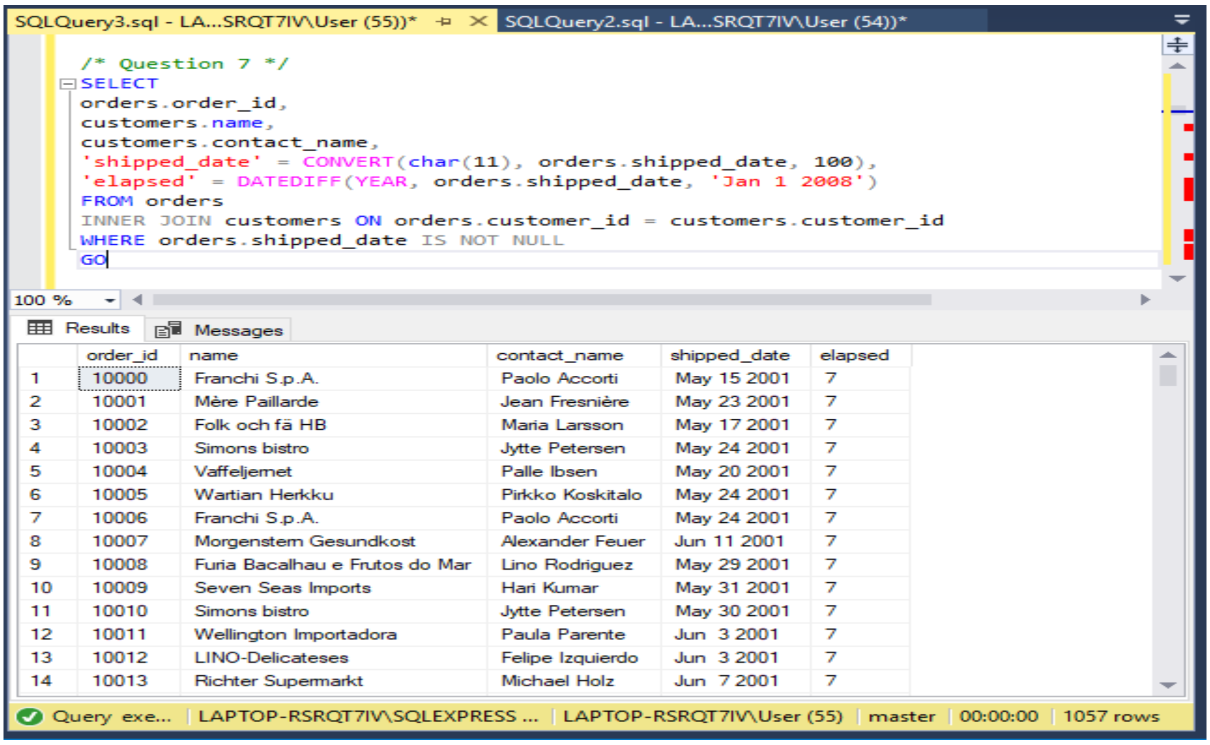
	supplier_name	products_name	reorder_level	quantity_in_stock
1	Armstrong Company	Queso Cabrales	30	22
2	Cadbury Products Ltd.	Ipoh Coffee	25	17
3	Cadbury Products Ltd.	Røgede sild	15	5
4	Campbell Company	Gnocchi di nonna Alice	30	21
5	Dare Manufacturer Ltd.	Scottish Longbreads	15	6
6	Dare Manufacturer Ltd.	Sir Rodney's Scones	5	3
7	Edward's Products Ltd.	Chang	25	17
8	Edward's Products Ltd.	Aniseed Syrup	25	13
9	Kaplan Ltd.	Nord-Ost Matjeshering	15	10
10	New Orlean's Spices Ltd.	Louisiana Hot Spiced Okra	20	4
11	Ovellette Manufacturer Company	Chocolade	25	15
12	South Harbour Products Ltd.	Maxilaku	15	10
13	South Harbour Products Ltd.	Wimmers gute Semmelknödel	30	22
14	St. Jean's Company	Gorgonzola Telino	20	0

Query execu... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (55) | master | 00:00:00 | 18 rows

7. Calculate the length in years from **January 1, 2008** and when an order was shipped where the shipped date is **not null**. Display the order id, and the shipped date from the orders table, the customer name, and the contact name from the customers table, and the length in years for each order. Display the shipped date in the format MMM DD YYYY. Order the result set by order id and the calculated years. The query should produce the result set listed below.

```
order_id     name                         contact_name                 shipped_date    elapsed
-----------  ---------------------------  ---------------------------  ------------    -------
10000        Franchi S.p.A.               Paolo Accorti                May 15 2001     7
10001        Mère Paillarde               Jean Fresnière               May 23 2001     7
10002        Folk och fä HB               Maria Larsson                May 17 2001     7
10003        Simons bistro                Jytte Petersen               May 24 2001     7
10004        Vaffeljernet                 Palle Ibsen                  May 20 2001     7
...
11066        White Clover Markets         Karl Jablonski               Mar 28 2004     4
11067        Drachenblut Delikatessen     Sven Ottlieb                 Mar 30 2004     4
11069        Tortuga Restaurante          Miguel Angel Paolino         Mar 30 2004     4

(1057 row(s) affected)
```

```sql
/* Question 7 */
SELECT
orders.order_id,
customers.name,
customers.contact_name,
'shipped_date' = CONVERT(char(11), orders.shipped_date, 100),
'elapsed' = DATEDIFF(YEAR, orders.shipped_date, 'Jan 1 2008')
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
WHERE orders.shipped_date IS NOT NULL
GO
```



```
/* Question 7 */
SELECT
orders.order_id,
customers.name,
customers.contact_name,
'shipped_date' = CONVERT(char(11), orders.shipped_date, 100),
'elapsed' = DATEDIFF(YEAR, orders.shipped_date, 'Jan 1 2008')
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
WHERE orders.shipped_date IS NOT NULL
GO
```

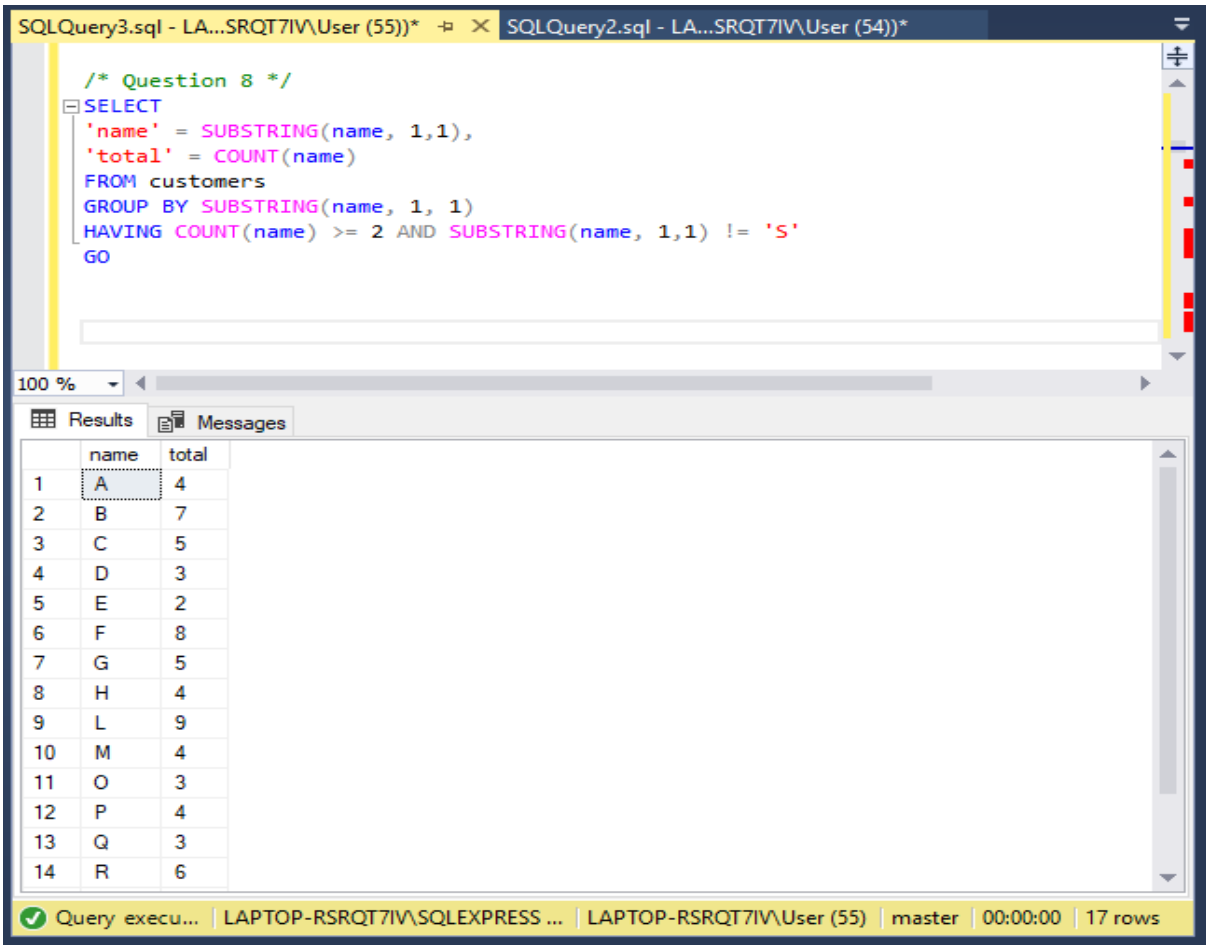
	order_id	name	contact_name	shipped_date	elapsed
1	10000	Franchi S.p.A.	Paolo Accorti	May 15 2001	7
2	10001	Mère Paillarde	Jean Fresnière	May 23 2001	7
3	10002	Folk och fä HB	Maria Larsson	May 17 2001	7
4	10003	Simons bistro	Jytte Petersen	May 24 2001	7
5	10004	Vaffeljernet	Palle Ibsen	May 20 2001	7
6	10005	Wartian Herkku	Pirkko Koskitalo	May 24 2001	7
7	10006	Franchi S.p.A.	Paolo Accorti	May 24 2001	7
8	10007	Morgenstern Gesundkost	Alexander Feuer	Jun 11 2001	7
9	10008	Furia Bacalhau e Frutos do Mar	Lino Rodriguez	May 29 2001	7
10	10009	Seven Seas Imports	Hari Kumar	May 31 2001	7
11	10010	Simons bistro	Jytte Petersen	May 30 2001	7
12	10011	Wellington Importadora	Paula Parente	Jun 3 2001	7
13	10012	LINO-Delicateses	Felipe Izquierdo	Jun 3 2001	7
14	10013	Richter Supermarkt	Michael Holz	Jun 7 2001	7

Query exe... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (55) | master | 00:00:00 | 1057 rows

8. List number of customers with names beginning with each letter of the alphabet. Ignore customers whose name begins with the letter **S**. Do not display the letter and count unless **at least two** customer's names begin with the letter. The query should produce the result set listed below.

```
name        total
------      -----------
A           4
B           7
C           5
D           3
E           2
...
T           6
V           3
W           5

(17 row(s) affected)
```

```
/* Question 8 */
SELECT
'name' = SUBSTRING(name, 1,1),
'total' = COUNT(name)
FROM customers
GROUP BY SUBSTRING(name, 1, 1)
HAVING COUNT(name) >= 2 AND SUBSTRING(name, 1, 1) != 'S'
GO
```


SQLQuery3.sql - LA...SRQT7IV\User (55))*
SQLQuery2.sql - LA...SRQT7IV\User (54))*
100 %
Results
Messages

```
/* Question 8 */
SELECT
'name' = SUBSTRING(name, 1,1),
'total' = COUNT(name)
FROM customers
GROUP BY SUBSTRING(name, 1, 1)
HAVING COUNT(name) >= 2 AND SUBSTRING(name, 1,1) != 'S'
GO
```

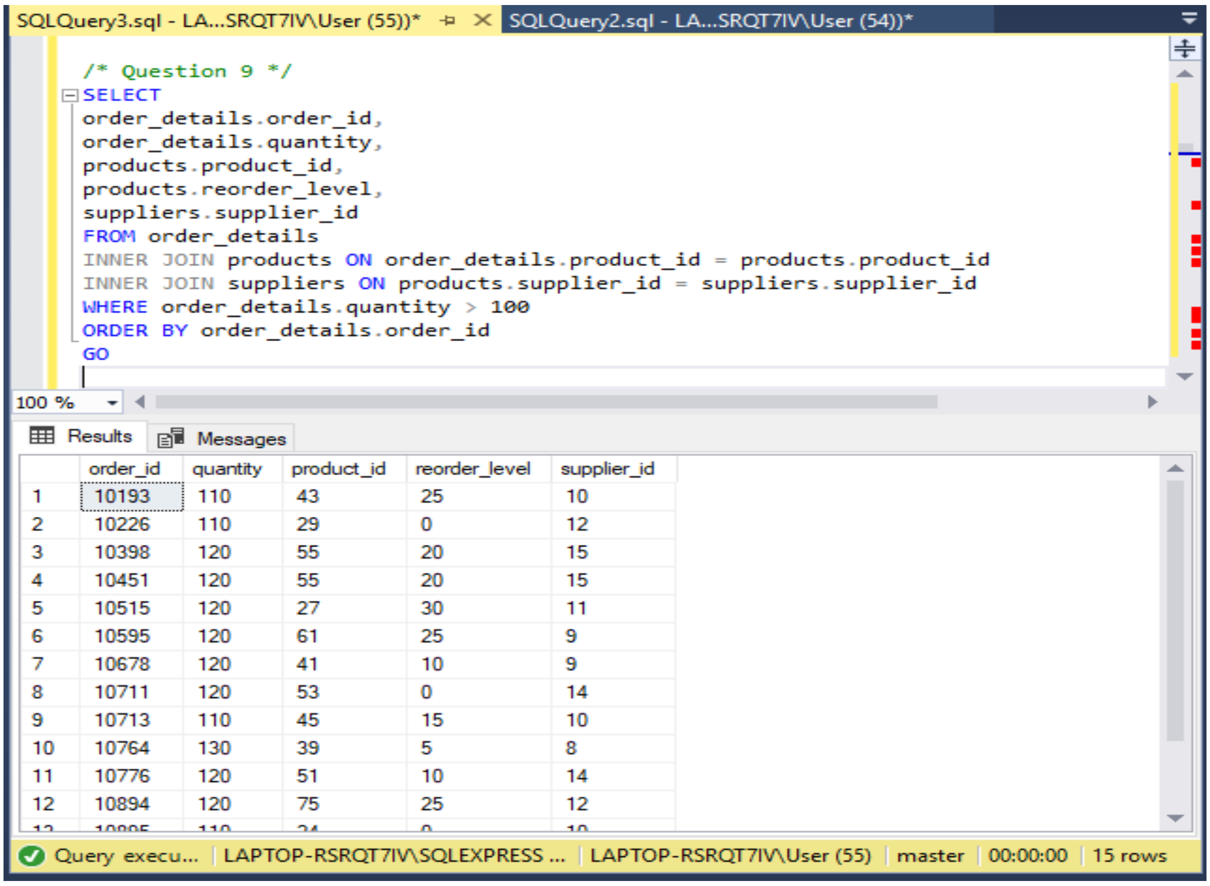
	name	total
1	A	4
2	B	7
3	C	5
4	D	3
5	E	2
6	F	8
7	G	5
8	H	4
9	L	9
10	M	4
11	O	3
12	P	4
13	Q	3
14	R	6

Query execu... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (55) | master | 00:00:00 | 17 rows

9. List the order details where the quantity is **greater than 100**. Display the order id and quantity from the order_details table, the product id and reorder level from the products table, and the supplier id from the suppliers table. Order the result set by the order id. The query should produce the result set listed below.

```
order_id      quantity        product_id      reorder_level     supplier_id
-----------   -----------     -------------   ----------------  -----------
10193         110             43              25                10
10226         110             29              0                 12
10398         120             55              20                15
10451         120             55              20                15
10515         120             27              30                11
...
10895         110             24              0                 10
11017         110             59              0                 8
11072         130             64              30                12

(15 row(s) affected)
```

```
/* Question 9 */
SELECT
order_details.order_id,
order_details.quantity,
products.product_id,
products.reorder_level,
suppliers.supplier_id
FROM order_details
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
WHERE order_details.quantity > 100
ORDER BY order_details.order_id
GO
```


SQLQuery3.sql - LA...SRQT7IV\User (55))*
SQLQuery2.sql - LA...SRQT7IV\User (54))*
100 %
Results
Messages

```
/* Question 9 */
SELECT
order_details.order_id,
order_details.quantity,
products.product_id,
products.reorder_level,
suppliers.supplier_id
FROM order_details
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
WHERE order_details.quantity > 100
ORDER BY order_details.order_id
GO
```

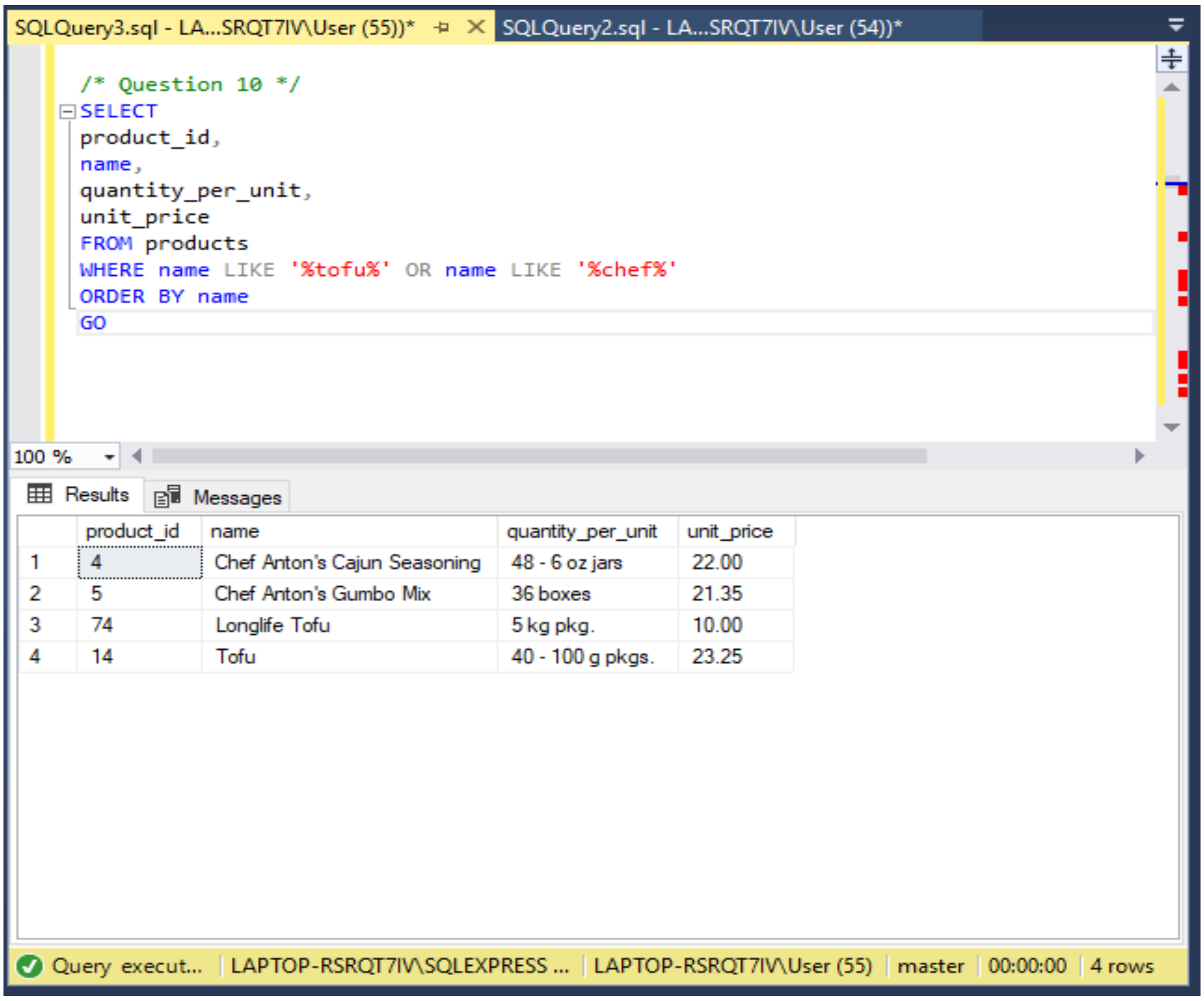
```
     order_id  quantity  product_id  reorder_level  supplier_id
1    10193     110       43          25             10
2    10226     110       29          0              12
3    10398     120       55          20             15
4    10451     120       55          20             15
5    10515     120       27          30             11
6    10595     120       61          25             9
7    10678     120       41          10             9
8    10711     120       53          0              14
9    10713     110       45          15             10
10   10764     130       39          5              8
11   10776     120       51          10             14
12   10894     120       75          25             12
```

```
Query execu... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (55) | master | 00:00:00 | 15 rows
```

10. List the products which contain **tofu** or **chef** in their name. Display the product id, product name, quantity per unit and unit price from the products table. Order the result set by product name. The query should produce the result set listed below.

```
product_id    name                                 quantity_per_unit      unit_price
------------- ------------------------------------ ---------------------- --------------
4             Chef Anton's Cajun Seasoning         48 - 6 oz jars         22.0000
5             Chef Anton's Gumbo Mix               36 boxes               21.3500
74            Longlife Tofu                        5 kg pkg.              10.0000
14            Tofu                                 40 - 100 g pkgs.       23.2500

(4 row(s) affected)
```

```
/* Question 10 */
SELECT
product_id,
name,
quantity_per_unit,
unit_price
FROM products
WHERE name LIKE '%tofu%' OR name LIKE '%chef%'
ORDER BY name
GO
```



```
/* Question 10 */
SELECT
product_id,
name,
quantity_per_unit,
unit_price
FROM products
WHERE name LIKE '%tofu%' OR name LIKE '%chef%'
ORDER BY name
GO
```

```
   product_id  name                        quantity_per_unit  unit_price
1  4           Chef Anton's Cajun Seasoning 48 - 6 oz jars    22.00
2  5           Chef Anton's Gumbo Mix      36 boxes           21.35
3  74          Longlife Tofu               5 kg pkg.          10.00
4  14          Tofu                        40 - 100 g pkgs.   23.25
```

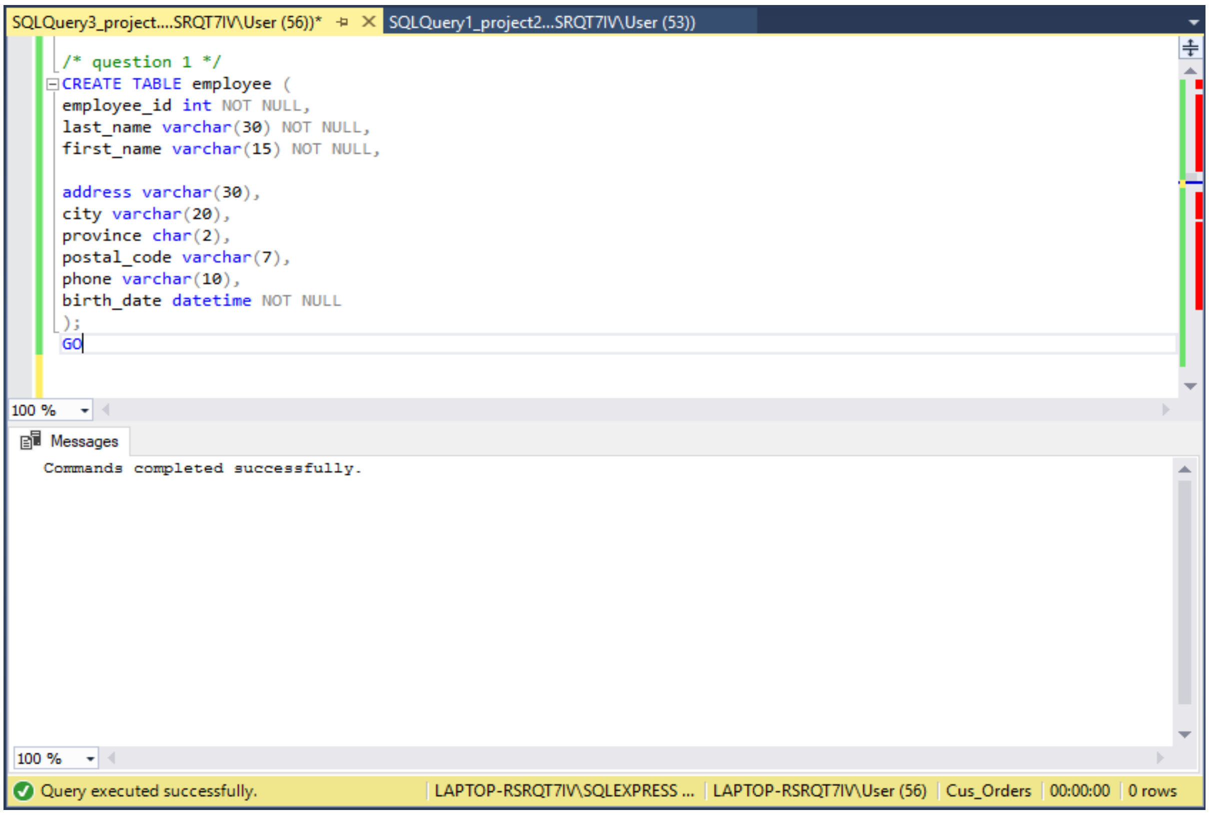
Query execut... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (55) | master | 00:00:00 | 4 rows

Part C - INSERT, UPDATE, DELETE and VIEWS Statements

1. Create an **employee** table with the following columns:

Column Name	*Data Type*	*Length*	*Null Values*
employee_id	int		No
last_name	varchar	30	No
first_name	varchar	15	No
address	varchar	30	
city	varchar	20	
province	char	2	
postal_code	varchar	7	
phone	varchar	10	
birth_date	datetime		No

```
/* Question 1 */
CREATE TABLE employee (
employee_id int NOT NULL,
last_name varchar(30) NOT NULL,
first_name varchar(15) NOT NULL,
address varchar(30),
city varchar(20),
province char(2),
postal_code varchar(7),
phone varchar(10),
birth_date datetime NOT NULL
);
GO
```



SQLQuery3_project....SRQT7IV\User (56))* SQLQuery1_project2...SRQT7IV\User (53))

```
/* question 1 */
CREATE TABLE employee (
employee_id int NOT NULL,
last_name varchar(30) NOT NULL,
first_name varchar(15) NOT NULL,

address varchar(30),
city varchar(20),
province char(2),
postal_code varchar(7),
phone varchar(10),
birth_date datetime NOT NULL
);
GO
```

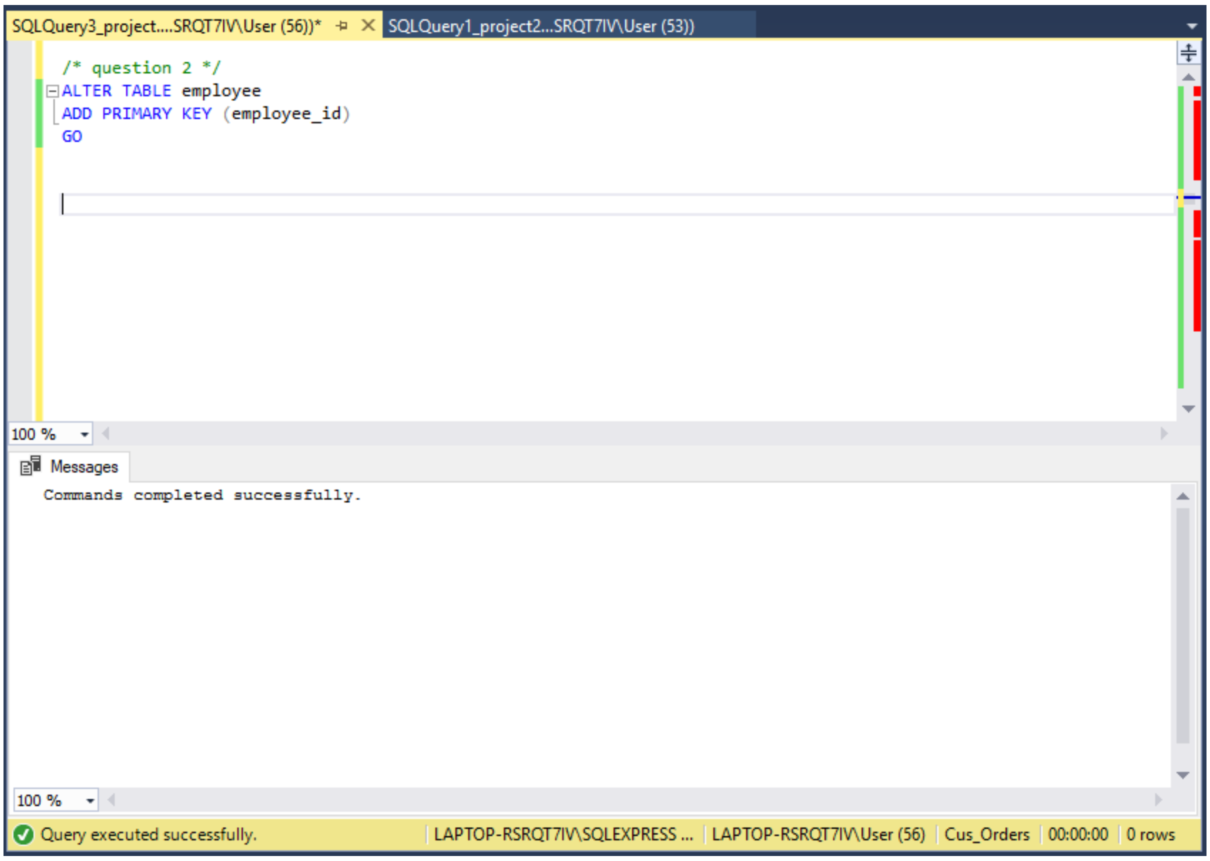
Messages

```
Commands completed successfully.
```

Query executed successfully. | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (56) | Cus_Orders | 00:00:00 | 0 rows

2. The **primary key** for the employee table should be the employee id.

```
/* Question 2 */
ALTER TABLE employee
ADD PRIMARY KEY (employee_id)
GO
```


SQLQuery3_project....SRQT7IV\User (56))*
SQLQuery1_project2...SRQT7IV\User (53))
100 %
Messages

```
/* question 2 */
ALTER TABLE employee
ADD PRIMARY KEY (employee_id)
GO
```

```
Commands completed successfully.
```

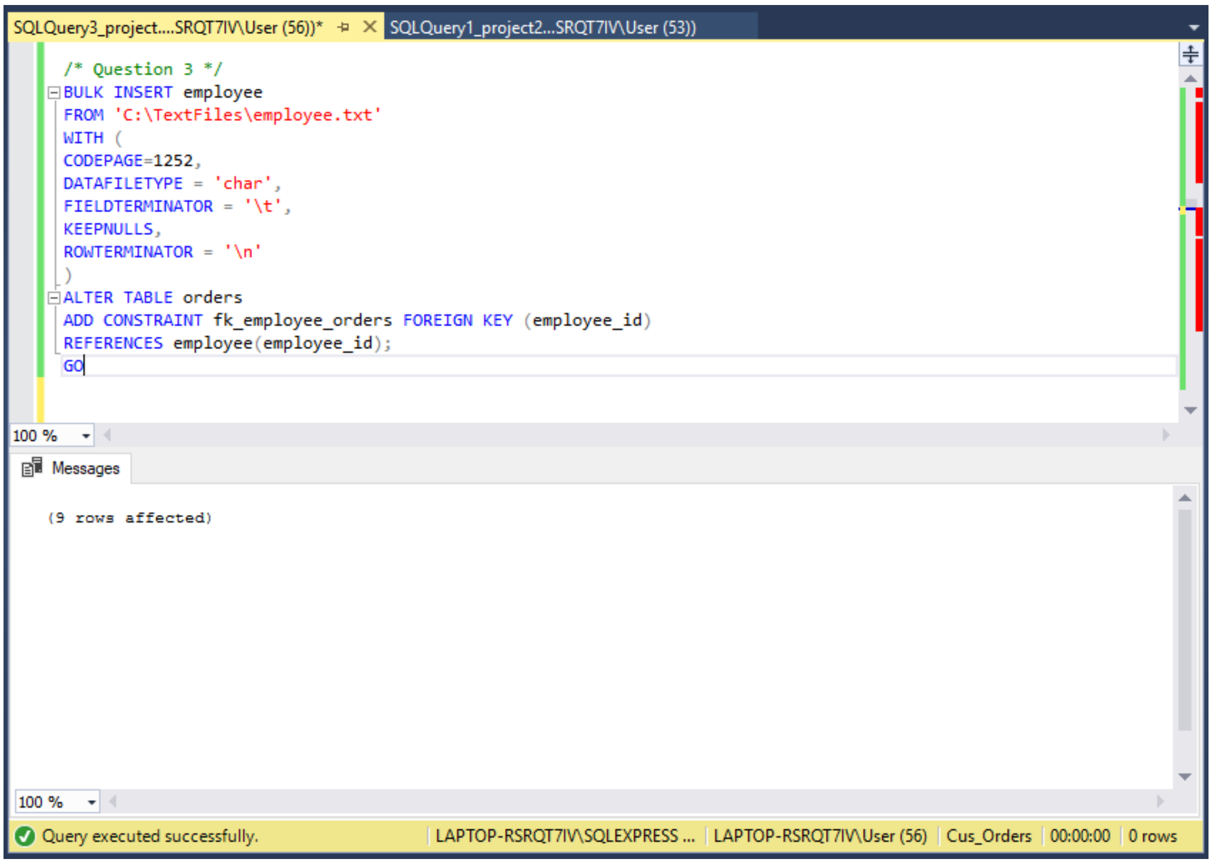
```
Query executed successfully.   LAPTOP-RSRQT7IV\SQLEXPRESS ...   LAPTOP-RSRQT7IV\User (56)   Cus_Orders   00:00:00   0 rows
```

3. Load the data into the employee table using the employee.txt file; **9** rows. In addition, **create the relationship** to enforce referential integrity between the employee and orders tables.

```
/* Question 3 */
BULK INSERT employee
FROM 'C:\TextFiles\employee.txt'
WITH (
CODEPAGE=1252,
DATAFILETYPE = 'char',
FIELDTERMINATOR = '\t',
KEEPNULLS,
ROWTERMINATOR = '\n'
)
ALTER TABLE orders
ADD CONSTRAINT fk_employee_orders FOREIGN KEY (employee_id)
REFERENCES employee(employee_id);
GO
```

4. Using the INSERT statement, add the shipper **Quick Express** to the shippers table.

```
/* Question 4 */
INSERT INTO shippers(name)
VALUES('Quick Express')
GO
```

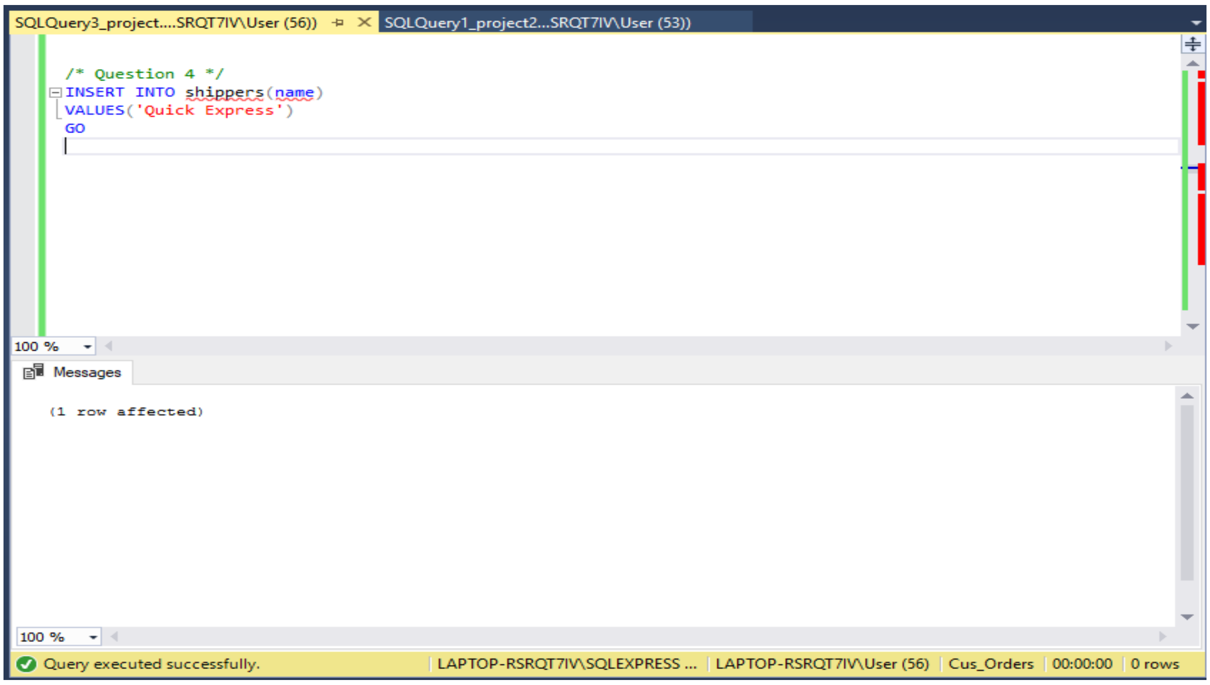

SQLQuery3_project....SRQT7IV\User (56))* SQLQuery1_project2...SRQT7IV\User (53))

```
/* Question 3 */
BULK INSERT employee
FROM 'C:\TextFiles\employee.txt'
WITH (
CODEPAGE=1252,
DATAFILETYPE = 'char',
FIELDTERMINATOR = '\t',
KEEPNULLS,
ROWTERMINATOR = '\n'
)
ALTER TABLE orders
ADD CONSTRAINT fk_employee_orders FOREIGN KEY (employee_id)
REFERENCES employee(employee_id);
GO
```

Messages

```
(9 rows affected)
```

Query executed successfully. | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (56) | Cus_Orders | 00:00:00 | 0 rows


SQLQuery3_project....SRQT7IV\User (56)) SQLQuery1_project2...SRQT7IV\User (53))

```
/* Question 4 */
INSERT INTO shippers(name)
VALUES('Quick Express')
GO
```

Messages

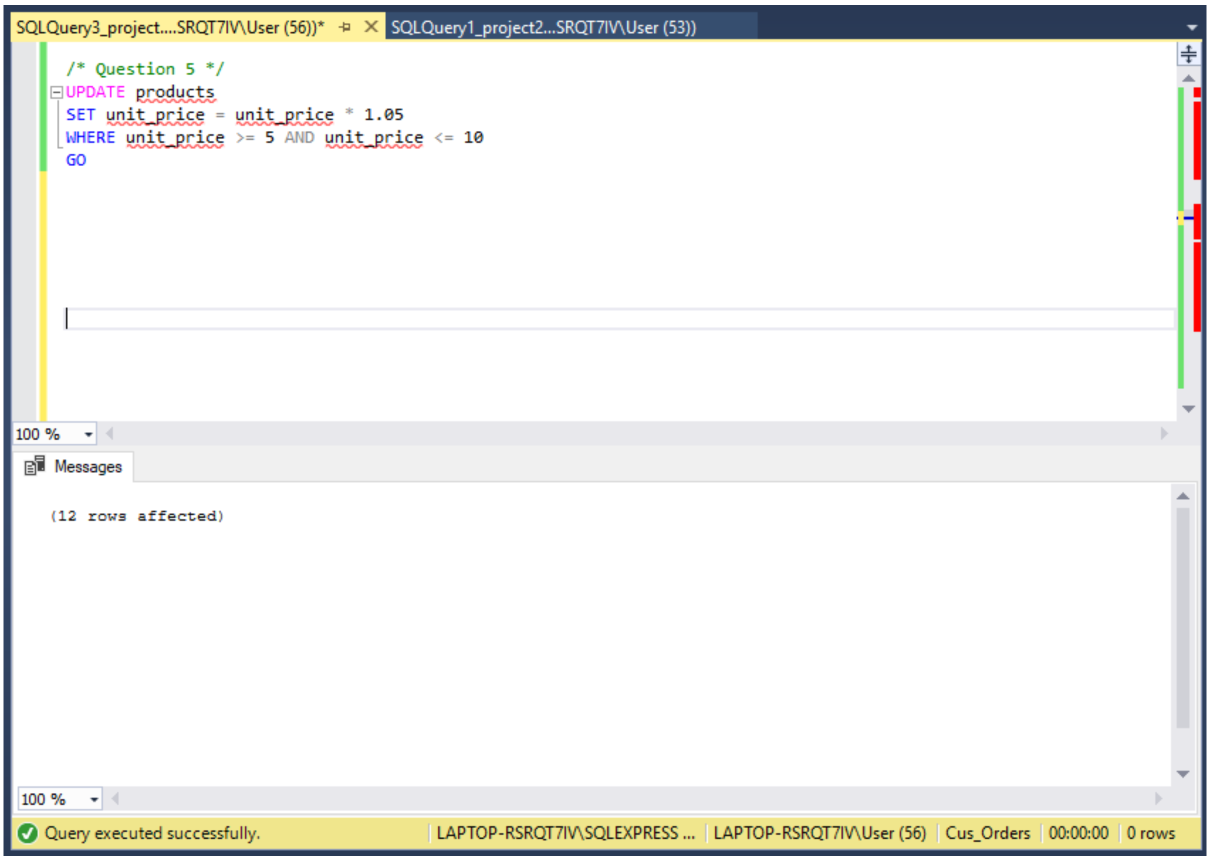
```
(1 row affected)
```

Query executed successfully. | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (56) | Cus_Orders | 00:00:00 | 0 rows

5. Using the UPDATE statement, increate the unit price in the products table of all rows with a current unit price between **$5.00** and **$10.00** by **5%**; 12 rows affected.

```
/* Question 5 */
UPDATE products
SET unit_price = unit_price * 1.05
WHERE unit_price >= 5 AND unit_price <= 10
```

```
GO
```



SQLQuery3_project....SRQT7IV\User (56))* | SQLQuery1_project2...SRQT7IV\User (53))

```
/* Question 5 */
UPDATE products
SET unit_price = unit_price * 1.05
WHERE unit_price >= 5 AND unit_price <= 10
GO
```

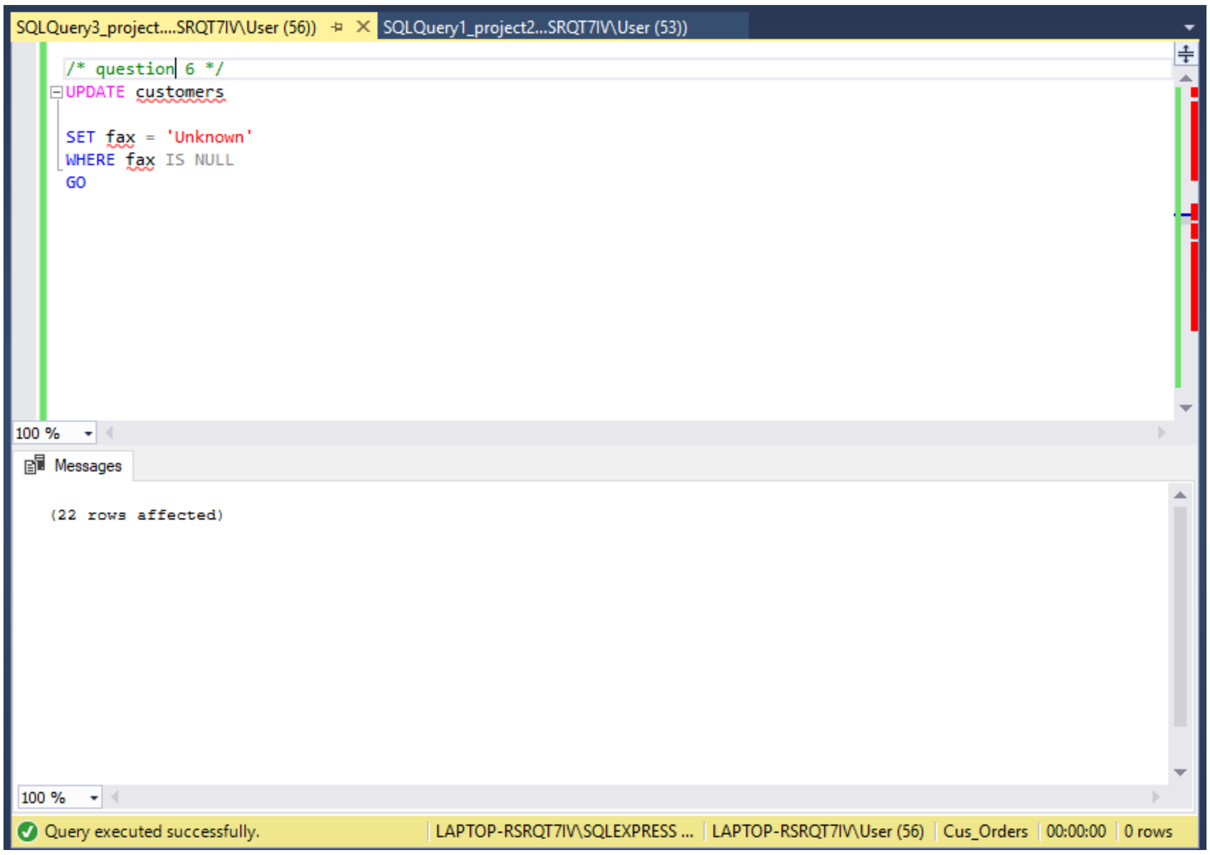
Messages

```
(12 rows affected)
```

Query executed successfully. | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (56) | Cus_Orders | 00:00:00 | 0 rows

6. Using the UPDATE statement, change the fax value to **Unknown** for all rows in the customers table where the current fax value is **NULL**; 22 rows affected.

```
/* Question 6 */
UPDATE customers

SET fax = 'Unknown'
WHERE fax IS NULL
GO
```


SQLQuery3_project....SRQT7IV\User (56)) SQLQuery1_project2...SRQT7IV\User (53))

```
/* question 6 */
UPDATE customers

SET fax = 'Unknown'
WHERE fax IS NULL
GO
```

Messages

```
(22 rows affected)
```

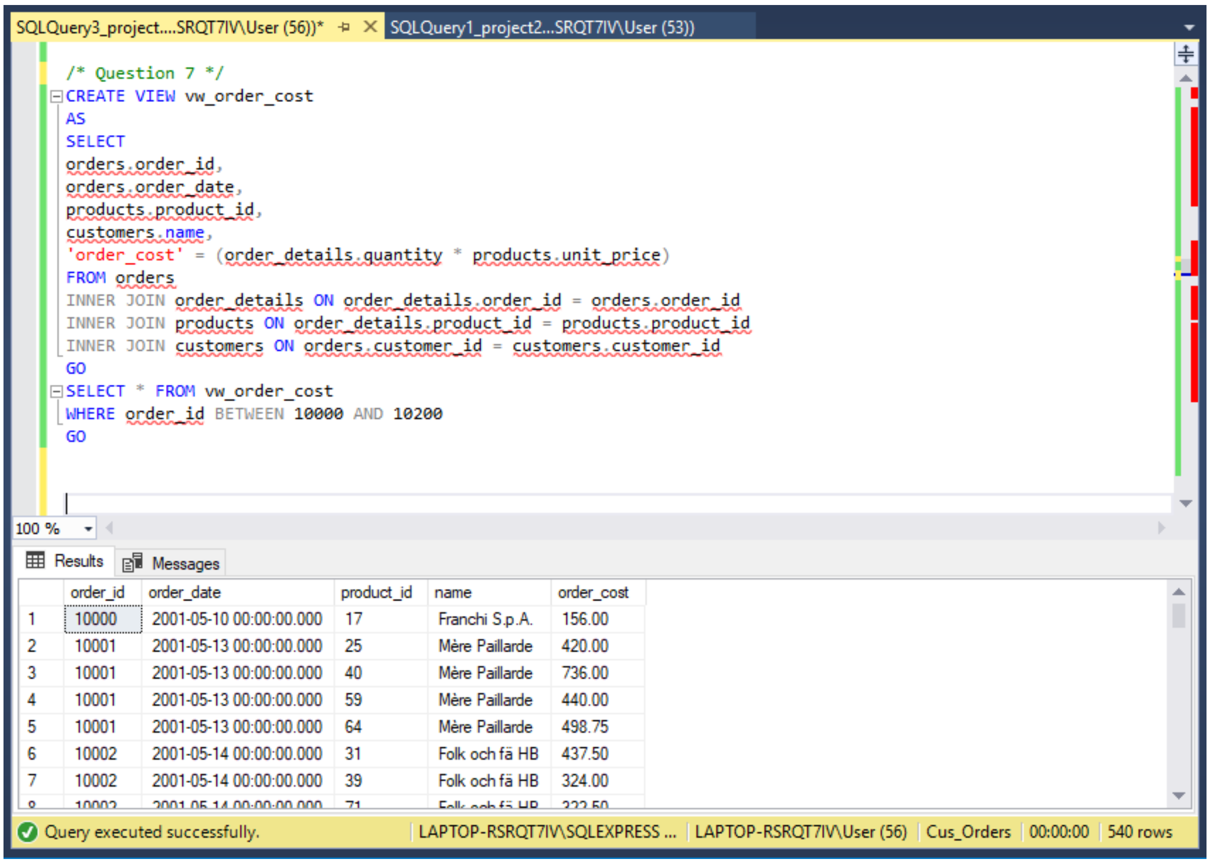
Query executed successfully. | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (56) | Cus_Orders | 00:00:00 | 0 rows

7. Create a view called **vw_order_cost** to list the cost of the orders. Display the order id and order_date from the orders table, the product id from the products table, the customer name from the customers tble, and the order cost. To calculate the cost of the orders, use the formula (order_details.quantity * products.unit_price). Run the view for the order ids between **10000** and **10200**. The view should produce the result set listed below.

```
order_id    order_date                    product_id    name                          order_cost
---------   ---------------------------   -----------   ---------------------------   -----------
10000       2001-05-10 00:00:00.000       17            Franchi S.p.A.                156.0000
10001       2001-05-13 00:00:00.000       25            Mère Paillarde                420.0000
10001       2001-05-13 00:00:00.000       40            Mère Paillarde                736.0000
10001       2001-05-13 00:00:00.000       59            Mère Paillarde                440.0000
10001       2001-05-13 00:00:00.000       64            Mère Paillarde                498.7500
...
10199       2002-03-27 00:00:00.000       3             Save-a-lot Markets            400.0000
10199       2002-03-27 00:00:00.000       39            Save-a-lot Markets            720.0000
10200       2002-03-30 00:00:00.000       11            Bólido Comidas preparadas     588.0000

(540 row(s) affected)
```

```
/* Question 7 */
CREATE VIEW vw_order_cost
AS
SELECT
orders.order_id,
orders.order_date,
products.product_id,
customers.name,
'order_cost' = (order_details.quantity * products.unit_price)
FROM orders
```

```
INNER JOIN order_details ON order_details.order_id = orders.order_id
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN customers ON orders.customer_id = customers.customer_id
GO
SELECT * FROM vw_order_cost
WHERE order_id BETWEEN 10000 AND 10200
GO
```



```
SQLQuery3_project....SRQT7IV\User (56))*    SQLQuery1_project2...SRQT7IV\User (53))

/* Question 7 */
CREATE VIEW vw_order_cost
AS
SELECT
orders.order_id,
orders.order_date,
products.product_id,
customers.name,
'order_cost' = (order_details.quantity * products.unit_price)
FROM orders
INNER JOIN order_details ON order_details.order_id = orders.order_id
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN customers ON orders.customer_id = customers.customer_id
GO
SELECT * FROM vw_order_cost
WHERE order_id BETWEEN 10000 AND 10200
GO
```

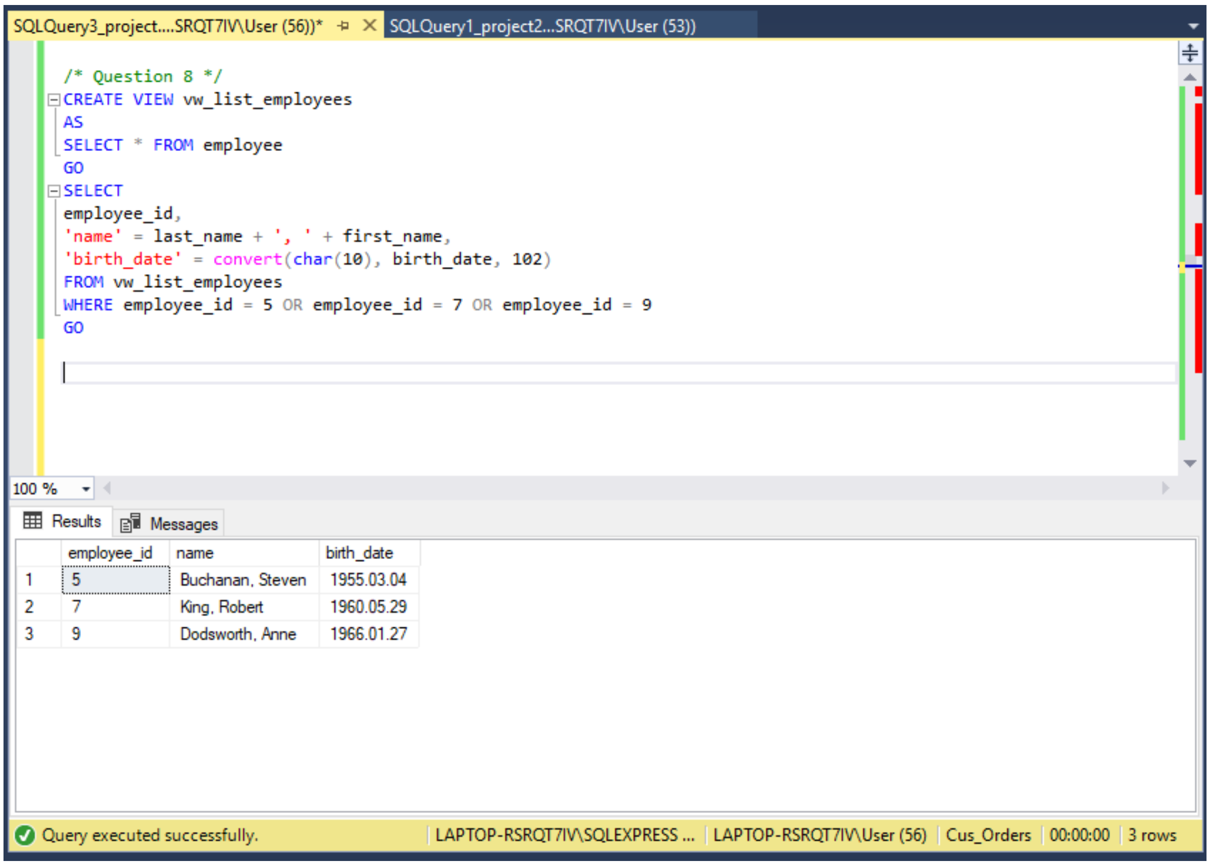
```
   order_id  order_date               product_id  name            order_cost
1  10000     2001-05-10 00:00:00.000  17          Franchi S.p.A.  156.00
2  10001     2001-05-13 00:00:00.000  25          Mère Paillarde  420.00
3  10001     2001-05-13 00:00:00.000  40          Mère Paillarde  736.00
4  10001     2001-05-13 00:00:00.000  59          Mère Paillarde  440.00
5  10001     2001-05-13 00:00:00.000  64          Mère Paillarde  498.75
6  10002     2001-05-14 00:00:00.000  31          Folk och fä HB  437.50
7  10002     2001-05-14 00:00:00.000  39          Folk och fä HB  324.00
```

```
Query executed successfully. | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (56) | Cus_Orders | 00:00:00 | 540 rows
```

8. Create a view called **vw_list_employees** to list all the employees and all the columns in the employee table. Run the view for employee ids **5**, **7**, and **9**. Display the employee id, last name, first name, and birth date. Format the name as last name followed by a comma and a space followed by the first name. Format the birth date as **YYYY.MM.DD**. The view should produce the result set listed below.

```
employee_id  name                              birth_date
-----------  --------------------------------  -----------
5            Buchanan, Steven                  1955.03.04
7            King, Robert                      1960.05.29
9            Dodsworth, Anne                   1966.01.27

(3 row(s) affected)
```

```
/* Question 8 */
CREATE VIEW vw_list_employees
AS
SELECT * FROM employee
GO
SELECT
employee_id,
'name' = last_name + ', ' + first_name,
'birth_date' = convert(char(10), birth_date, 102)
FROM vw_list_employees
WHERE employee_id = 5 OR employee_id = 7 OR employee_id = 9
GO
```



SQLQuery3_project....SRQT7IV\User (56))* SQLQuery1_project2...SRQT7IV\User (53))

```
/* Question 8 */
CREATE VIEW vw_list_employees
AS
SELECT * FROM employee
GO
SELECT
employee_id,
'name' = last_name + ', ' + first_name,
'birth_date' = convert(char(10), birth_date, 102)
FROM vw_list_employees
WHERE employee_id = 5 OR employee_id = 7 OR employee_id = 9
GO
```

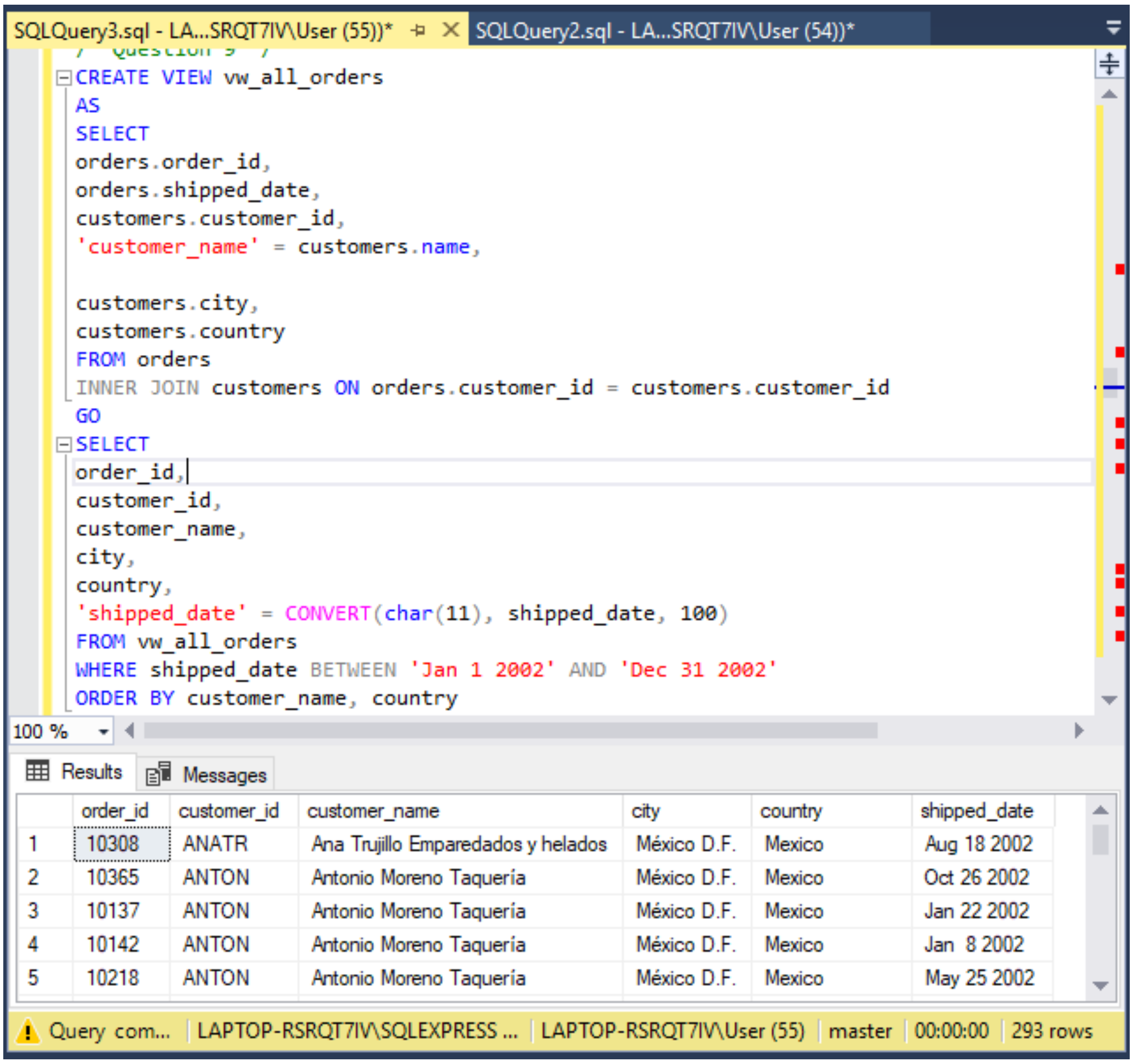
	employee_id	name	birth_date
1	5	Buchanan, Steven	1955.03.04
2	7	King, Robert	1960.05.29
3	9	Dodsworth, Anne	1966.01.27

Query executed successfully. | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (56) | Cus_Orders | 00:00:00 | 3 rows

9. Create a view called **vw_all_orders** to list all the orders. Display the order id and shipped date from the orders table, and the customer id, name, city, and country from the customers table. Run the view for orders shipped from **January 1, 2002** and **December 31, 2002**, formatting the shipped date as **MON DD YYYY**. Order the result set by customer name and country. The view should produce the result set listed below.

```
order_id  customer_id  customer_name                               city              country          shipped_date
--------- -----------  ------------------------------------------  ----------------  ---------------  ------------
10308     ANATR        Ana Trujillo Emparedados y helados          México D.F.       Mexico           Aug 18 2002
10365     ANTON        Antonio Moreno Taquería                     México D.F.       Mexico           Oct 26 2002
10137     ANTON        Antonio Moreno Taquería                     México D.F.       Mexico           Jan 22 2002
10142     ANTON        Antonio Moreno Taquería                     México D.F.       Mexico           Jan  8 2002
10218     ANTON        Antonio Moreno Taquería                     México D.F.       Mexico           May 25 2002
...
10344     WHITC        White Clover Markets                        Seattle           United States    Sep 29 2002
10269     WHITC        White Clover Markets                        Seattle           United States    Jul 3 2002
10374     WOLZA        Wolski  Zajazd                              Warszawa          Poland           Nov 2 2002

(293 row(s) affected)
```

```
/* Question 9 */
CREATE VIEW vw_all_orders
AS
SELECT
orders.order_id,
orders.shipped_date,
customers.customer_id,
'customer_name' = customers.name,

customers.city,
customers.country
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
GO
SELECT
order_id,
customer_id,
customer_name,
city,
country,
'shipped_date' = CONVERT(char(11), shipped_date, 100)
FROM vw_all_orders
WHERE shipped_date BETWEEN 'Jan 1 2002' AND 'Dec 31 2002'
ORDER BY customer_name, country
GO
```



SQLQuery3.sql - LA...SRQT7IV\User (55))* SQLQuery2.sql - LA...SRQT7IV\User (54))*

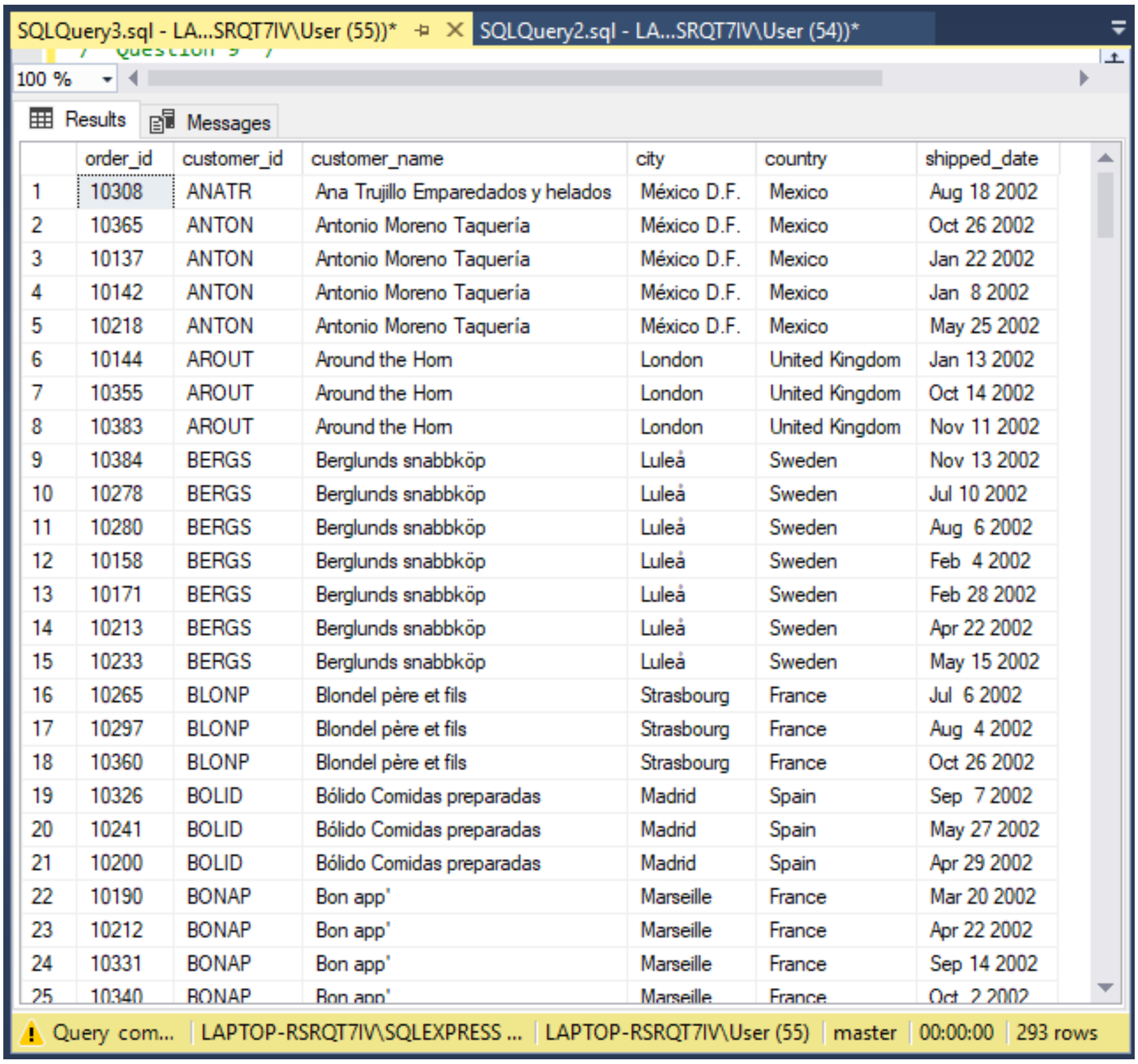
```
/* Question 9 */
CREATE VIEW vw_all_orders
AS
SELECT
orders.order_id,
orders.shipped_date,
customers.customer_id,
'customer_name' = customers.name,

customers.city,
customers.country
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
GO
SELECT
order_id,
customer_id,
customer_name,
city,
country,
'shipped_date' = CONVERT(char(11), shipped_date, 100)
FROM vw_all_orders
WHERE shipped_date BETWEEN 'Jan 1 2002' AND 'Dec 31 2002'
ORDER BY customer_name, country
```

100 %

Results | Messages

	order_id	customer_id	customer_name	city	country	shipped_date
1	10308	ANATR	Ana Trujillo Emparedados y helados	México D.F.	Mexico	Aug 18 2002
2	10365	ANTON	Antonio Moreno Taquería	México D.F.	Mexico	Oct 26 2002
3	10137	ANTON	Antonio Moreno Taquería	México D.F.	Mexico	Jan 22 2002
4	10142	ANTON	Antonio Moreno Taquería	México D.F.	Mexico	Jan 8 2002
5	10218	ANTON	Antonio Moreno Taquería	México D.F.	Mexico	May 25 2002

Query com... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (55) | master | 00:00:00 | 293 rows



SQLQuery3.sql - LA...SRQT7IV\User (55))* | SQLQuery2.sql - LA...SRQT7IV\User (54))*

100 %

Results | Messages

	order_id	customer_id	customer_name	city	country	shipped_date
1	10308	ANATR	Ana Trujillo Emparedados y helados	México D.F.	Mexico	Aug 18 2002
2	10365	ANTON	Antonio Moreno Taquería	México D.F.	Mexico	Oct 26 2002
3	10137	ANTON	Antonio Moreno Taquería	México D.F.	Mexico	Jan 22 2002
4	10142	ANTON	Antonio Moreno Taquería	México D.F.	Mexico	Jan 8 2002
5	10218	ANTON	Antonio Moreno Taquería	México D.F.	Mexico	May 25 2002
6	10144	AROUT	Around the Horn	London	United Kingdom	Jan 13 2002
7	10355	AROUT	Around the Horn	London	United Kingdom	Oct 14 2002
8	10383	AROUT	Around the Horn	London	United Kingdom	Nov 11 2002
9	10384	BERGS	Berglunds snabbköp	Luleå	Sweden	Nov 13 2002
10	10278	BERGS	Berglunds snabbköp	Luleå	Sweden	Jul 10 2002
11	10280	BERGS	Berglunds snabbköp	Luleå	Sweden	Aug 6 2002
12	10158	BERGS	Berglunds snabbköp	Luleå	Sweden	Feb 4 2002
13	10171	BERGS	Berglunds snabbköp	Luleå	Sweden	Feb 28 2002
14	10213	BERGS	Berglunds snabbköp	Luleå	Sweden	Apr 22 2002
15	10233	BERGS	Berglunds snabbköp	Luleå	Sweden	May 15 2002
16	10265	BLONP	Blondel père et fils	Strasbourg	France	Jul 6 2002
17	10297	BLONP	Blondel père et fils	Strasbourg	France	Aug 4 2002
18	10360	BLONP	Blondel père et fils	Strasbourg	France	Oct 26 2002
19	10326	BOLID	Bólido Comidas preparadas	Madrid	Spain	Sep 7 2002
20	10241	BOLID	Bólido Comidas preparadas	Madrid	Spain	May 27 2002
21	10200	BOLID	Bólido Comidas preparadas	Madrid	Spain	Apr 29 2002
22	10190	BONAP	Bon app'	Marseille	France	Mar 20 2002
23	10212	BONAP	Bon app'	Marseille	France	Apr 22 2002
24	10331	BONAP	Bon app'	Marseille	France	Sep 14 2002
25	10340	BONAP	Bon app'	Marseille	France	Oct 2 2002

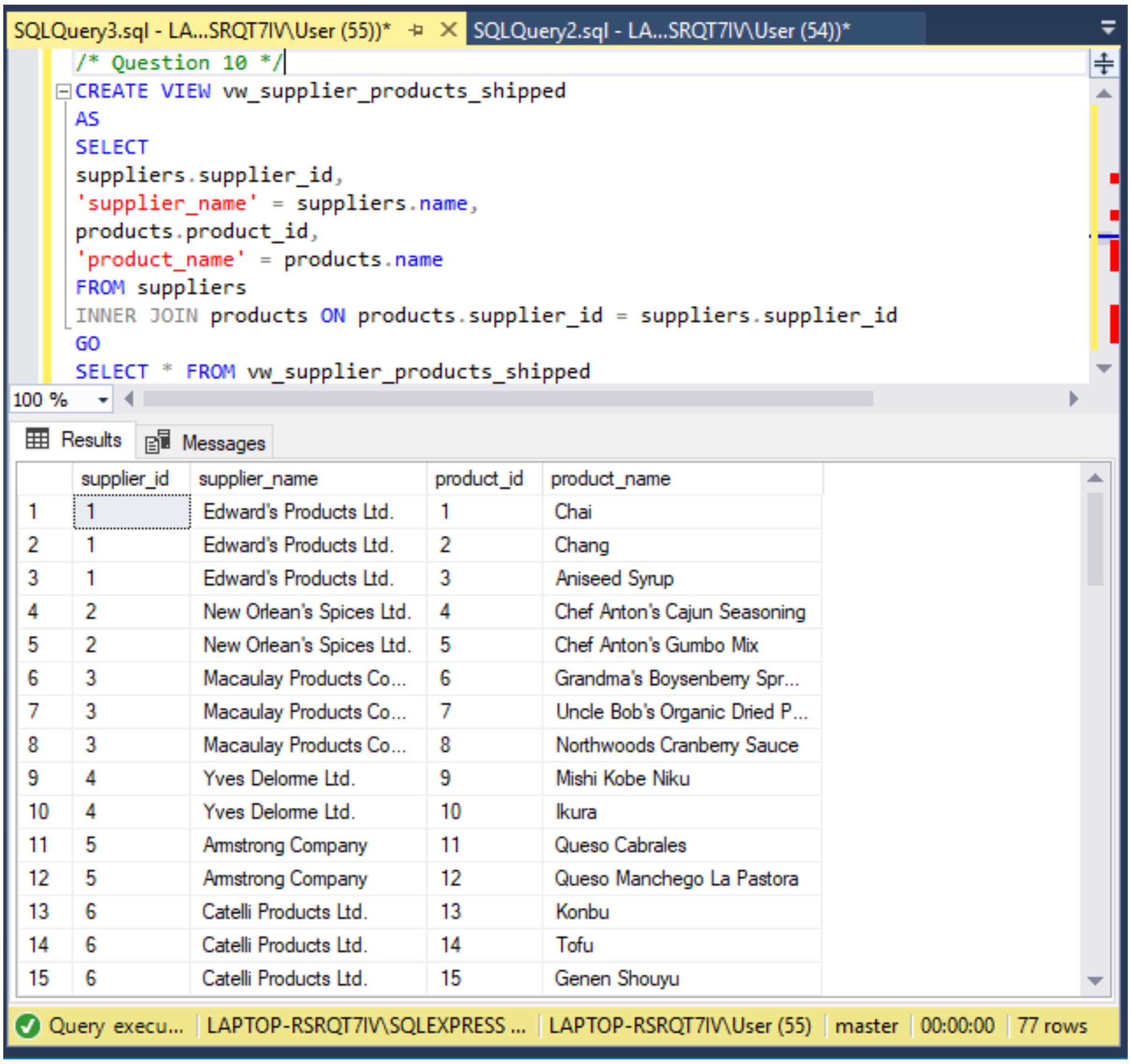
Query com... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (55) | master | 00:00:00 | 293 rows

10. Create a view listing the suppliers and the items they have shipped. Display the supplier id and name from the suppliers table, and the product id and name from the products table. Run the view. The view should produce the result set listed below, *although not necessarily in the same order.*

```
supplier_id   supplier_name                          product_id    product_name
-----------   ------------------------------------   -----------   ----------------------------
9             Silver Spring Wholesale Market          23            Tunnbröd
11            Ovellette Manufacturer Company          46            Spegesild
15            Campbell Company                        69            Gudbrandsdalsost
12            South Harbour Products Ltd.             77            Original Frankfurter grüne Soße
14            St. Jean's Company                      31            Gorgonzola Telino
...
7             Steveston Export Company                63            Vegie-spread
3             Macaulay Products Company               8             Northwoods Cranberry Sauce
15            Campbell Company                        55            Pâté chinois

(77 row(s) affected)
```

```sql
/* Question 10 */
CREATE VIEW vw_supplier_products_shipped
```

```
AS
SELECT
suppliers.supplier_id,
'supplier_name' = suppliers.name,
products.product_id,
'product_name' = products.name
FROM suppliers
INNER JOIN products ON products.supplier_id = suppliers.supplier_id
GO
SELECT * FROM vw_supplier_products_shipped
GO
```



SQLQuery3.sql - LA...SRQT7IV\User (55))* SQLQuery2.sql - LA...SRQT7IV\User (54))*

```
/* Question 10 */
CREATE VIEW vw_supplier_products_shipped
AS
SELECT
suppliers.supplier_id,
'supplier_name' = suppliers.name,
products.product_id,
'product_name' = products.name
FROM suppliers
INNER JOIN products ON products.supplier_id = suppliers.supplier_id
GO
SELECT * FROM vw_supplier_products_shipped
```

	supplier_id	supplier_name	product_id	product_name
1	1	Edward's Products Ltd.	1	Chai
2	1	Edward's Products Ltd.	2	Chang
3	1	Edward's Products Ltd.	3	Aniseed Syrup
4	2	New Orlean's Spices Ltd.	4	Chef Anton's Cajun Seasoning
5	2	New Orlean's Spices Ltd.	5	Chef Anton's Gumbo Mix
6	3	Macaulay Products Co...	6	Grandma's Boysenberry Spr...
7	3	Macaulay Products Co...	7	Uncle Bob's Organic Dried P...
8	3	Macaulay Products Co...	8	Northwoods Cranberry Sauce
9	4	Yves Delorme Ltd.	9	Mishi Kobe Niku
10	4	Yves Delorme Ltd.	10	Ikura
11	5	Armstrong Company	11	Queso Cabrales
12	5	Armstrong Company	12	Queso Manchego La Pastora
13	6	Catelli Products Ltd.	13	Konbu
14	6	Catelli Products Ltd.	14	Tofu
15	6	Catelli Products Ltd.	15	Genen Shouyu

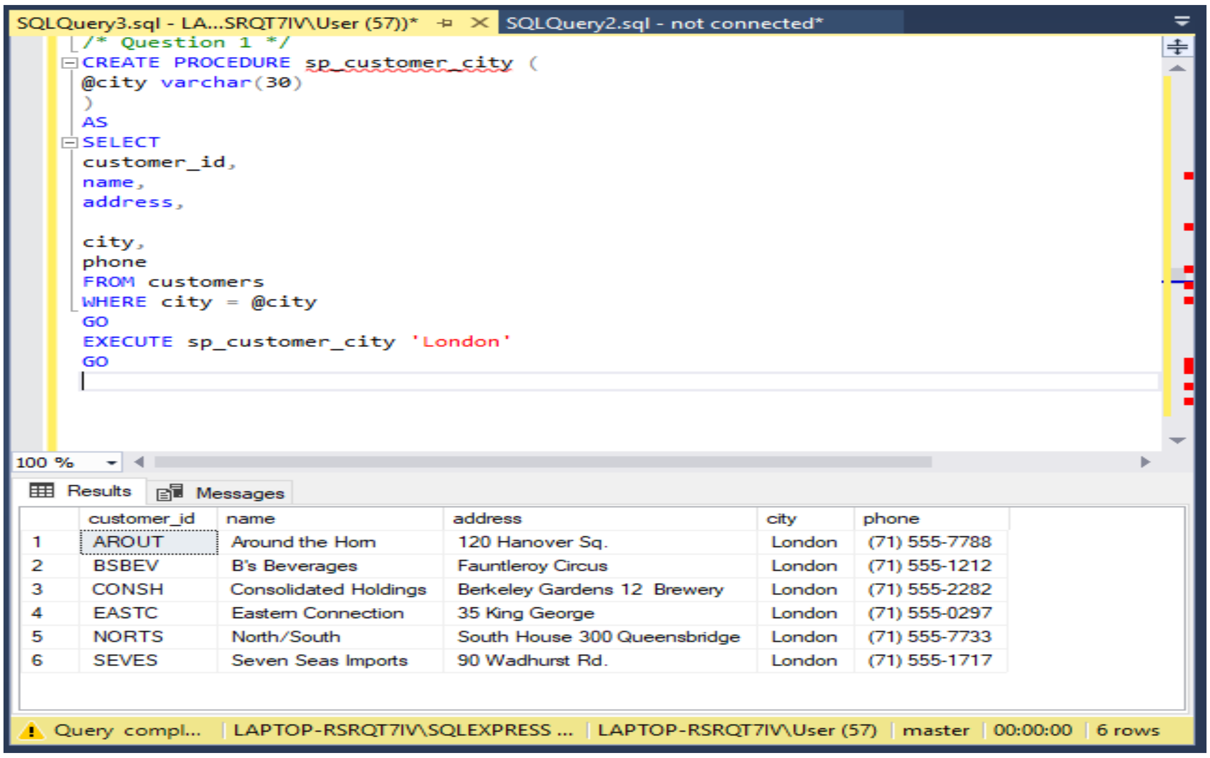
Query execu... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (55) | master | 00:00:00 | 77 rows

Part D - Stored Procedures and Triggers

1. Create a stored procedure called **sp_customer_city** displaying the customers living in a particular city. The **city** will be an **input parameter** for the stored procedure. Display the customer id, name, address, city and phone from the customers table. Run the stored procedure displaying customers living in **London**. The stored procedure should produce the result set listed below.

```
customer_id   name                    address                            city     phone
-----------   ---------------------   ----------  ----------------------------   ------   ----------
AROUT         Around the Horn         120 Hanover Sq.                    London   (71) 555-7788
BSBEV         B's Beverages           Fauntleroy Circus                  London   (71) 555-1212
CONSH         Consolidated Holdings   Berkeley Gardens 12  Brewery       London   (71) 555-2282
EASTC         Eastern Connection      35 King George                     London   (71) 555-0297
NORTS         North/South             South House 300 Queensbridge       London   (71) 555-7733
SEVES         Seven Seas Imports      90 Wadhurst Rd.                    London   (71) 555-1717

(6 row(s) affected)
```

```sql
/* Question 1 */
CREATE PROCEDURE sp_customer_city (
@city varchar(30)
)
AS
SELECT
customer_id,
name,
address,

city,
phone
FROM customers
WHERE city = @city
GO
EXECUTE sp_customer_city 'London'
GO
```



SQLQuery3.sql - LA...SRQT7IV\User (57))* SQLQuery2.sql - not connected*

```sql
/* Question 1 */
CREATE PROCEDURE sp_customer_city (
@city varchar(30)
)
AS
SELECT
customer_id,
name,
address,

city,
phone
FROM customers
WHERE city = @city
GO
EXECUTE sp_customer_city 'London'
GO
```

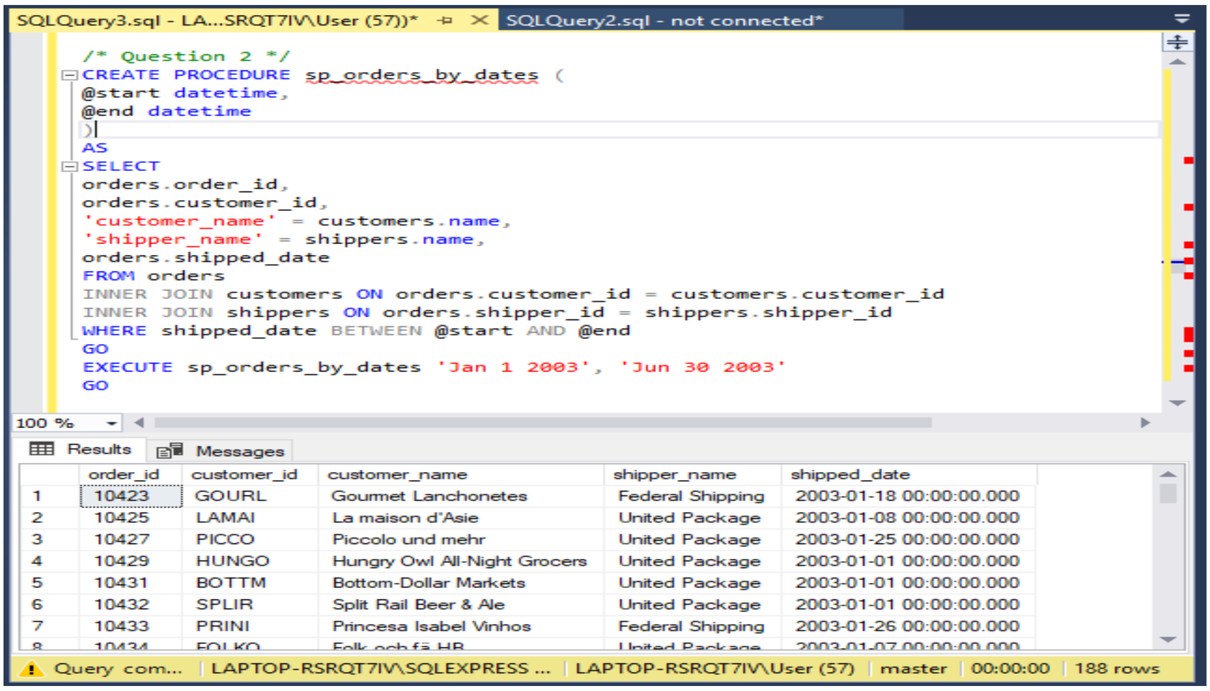
	customer_id	name	address	city	phone
1	AROUT	Around the Horn	120 Hanover Sq.	London	(71) 555-7788
2	BSBEV	B's Beverages	Fauntleroy Circus	London	(71) 555-1212
3	CONSH	Consolidated Holdings	Berkeley Gardens 12 Brewery	London	(71) 555-2282
4	EASTC	Eastern Connection	35 King George	London	(71) 555-0297
5	NORTS	North/South	South House 300 Queensbridge	London	(71) 555-7733
6	SEVES	Seven Seas Imports	90 Wadhurst Rd.	London	(71) 555-1717

Query compl... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (57) | master | 00:00:00 | 6 rows

2. Create a stored procedure called **sp_orders_by_dates** displaying the orders shipped between particular dates. The **start** and **end** date will be **input parameters** for the stored procedure. Display the order id, customer id, and shipped date from the orders table, the customer name from the customer table, and the shipper name from the shippers table. Run the stored procedure displaying orders from **January 1, 2003** to **June 30, 2003**. The stored procedure should produce the result set listed below.

```
order_id    customer_id    customer_name                      shipper_name       shipped_date
----------  -------------  ---------------------------------  -----------------  -----------------------
10423       GOURL          Gourmet Lanchonetes                Federal Shipping   2003-01-18 00:00:00.000
10425       LAMAI          La maison d'Asie                   United Package     2003-01-08 00:00:00.000
10427       PICCO          Piccolo und mehr                   United Package     2003-01-25 00:00:00.000
10429       HUNGO          Hungry Owl All-Night Grocers       United Package     2003-01-01 00:00:00.000
10431       BOTTM          Bottom-Dollar Markets              United Package     2003-01-01 00:00:00.000
...
10615       WILMK          Wilman Kala                        Federal Shipping   2003-06-30 00:00:00.000
10616       GREAL          Great Lakes Food Market            United Package     2003-06-29 00:00:00.000
10617       GREAL          Great Lakes Food Market            United Package     2003-06-28 00:00:00.000

(188 row(s) affected)
```

```sql
/* Question 2 */
CREATE PROCEDURE sp_orders_by_dates (
@start datetime,
@end datetime
)
AS
SELECT
orders.order_id,
orders.customer_id,
'customer_name' = customers.name,
'shipper_name' = shippers.name,
orders.shipped_date
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
INNER JOIN shippers ON orders.shipper_id = shippers.shipper_id
WHERE shipped_date BETWEEN @start AND @end
GO
EXECUTE sp_orders_by_dates 'Jan 1 2003', 'Jun 30 2003'
GO
```


SQLQuery3.sql - LA...SRQT7IV\User (57))*
SQLQuery2.sql - not connected*
100 %
Results
Messages

```sql
/* Question 2 */
CREATE PROCEDURE sp_orders_by_dates (
@start datetime,
@end datetime
)
AS
SELECT
orders.order_id,
orders.customer_id,
'customer_name' = customers.name,
'shipper_name' = shippers.name,
orders.shipped_date
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
INNER JOIN shippers ON orders.shipper_id = shippers.shipper_id
WHERE shipped_date BETWEEN @start AND @end
GO
EXECUTE sp_orders_by_dates 'Jan 1 2003', 'Jun 30 2003'
GO
```

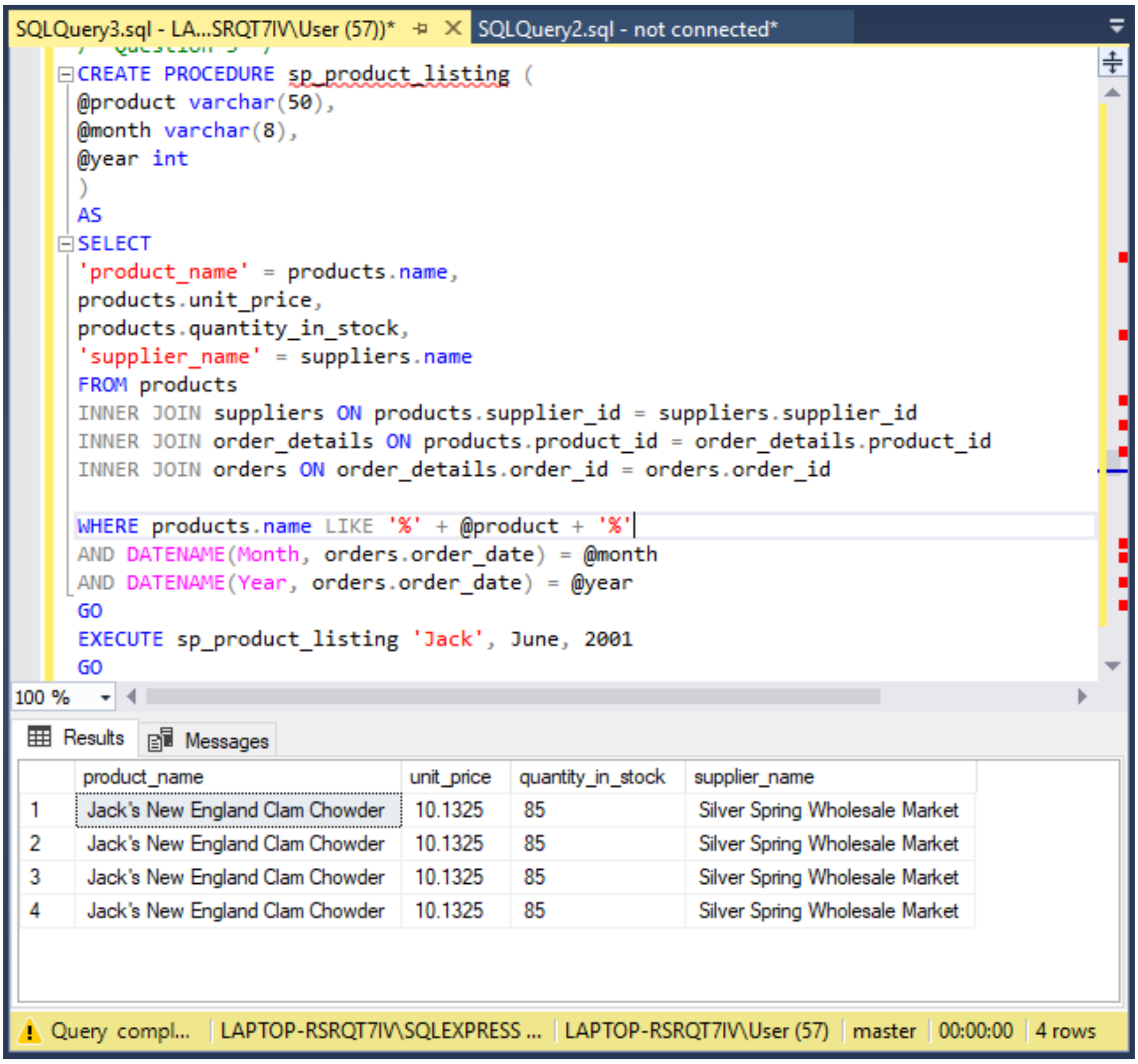
	order_id	customer_id	customer_name	shipper_name	shipped_date
1	10423	GOURL	Gourmet Lanchonetes	Federal Shipping	2003-01-18 00:00:00.000
2	10425	LAMAI	La maison d'Asie	United Package	2003-01-08 00:00:00.000
3	10427	PICCO	Piccolo und mehr	United Package	2003-01-25 00:00:00.000
4	10429	HUNGO	Hungry Owl All-Night Grocers	United Package	2003-01-01 00:00:00.000
5	10431	BOTTM	Bottom-Dollar Markets	United Package	2003-01-01 00:00:00.000
6	10432	SPLIR	Split Rail Beer & Ale	United Package	2003-01-01 00:00:00.000
7	10433	PRINI	Princesa Isabel Vinhos	Federal Shipping	2003-01-26 00:00:00.000
8	10434	FOLKO	Folk och fä HB	United Package	2003-01-07 00:00:00.000

Query com... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (57) | master | 00:00:00 | 188 rows

3. Create a stored procedure called **sp_product_listing** listing a specified product ordered during a specified month and year. The **product** and the **month** and **year** will be **input parameters** for the stored procedure. Display the product name, unit price, and quantity in stock from the products table, and the supplier name from the suppliers table. Run the stored procedure displaying a product name containing **Jack** and the month of the order date is **June** and the year is **2001**. The stored procedure should produce the result set listed below.

```
product_name                              unit_price    quantity_in_stock  supplier_name
----------------------------------------  ------------  -----------------  ----------------------------------
Jack's New England Clam Chowder           10.1325       85                 Silver Spring Wholesale Market
Jack's New England Clam Chowder           10.1325       85                 Silver Spring Wholesale Market
Jack's New England Clam Chowder           10.1325       85                 Silver Spring Wholesale Market
Jack's New England Clam Chowder           10.1325       85                 Silver Spring Wholesale Market

(4 row(s) affected)
```

```sql
/* Question 3 */
CREATE PROCEDURE sp_product_listing (
@product varchar(50),
@month varchar(8),
@year int
)
AS
SELECT
'product_name' = products.name,
products.unit_price,
products.quantity_in_stock,
'supplier_name' = suppliers.name
FROM products
INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
INNER JOIN order_details ON products.product_id = order_details.product_id
INNER JOIN orders ON order_details.order_id = orders.order_id

WHERE products.name LIKE '%' + @product + '%'
AND DATENAME(Month, orders.order_date) = @month
AND DATENAME(Year, orders.order_date) = @year
GO
EXECUTE sp_product_listing 'Jack', June, 2001
GO
```



```
CREATE PROCEDURE sp_product_listing (
@product varchar(50),
@month varchar(8),
@year int
)
AS
SELECT
'product_name' = products.name,
products.unit_price,
products.quantity_in_stock,
'supplier_name' = suppliers.name
FROM products
INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
INNER JOIN order_details ON products.product_id = order_details.product_id
INNER JOIN orders ON order_details.order_id = orders.order_id

WHERE products.name LIKE '%' + @product + '%'
AND DATENAME(Month, orders.order_date) = @month
AND DATENAME(Year, orders.order_date) = @year
GO
EXECUTE sp_product_listing 'Jack', June, 2001
GO
```

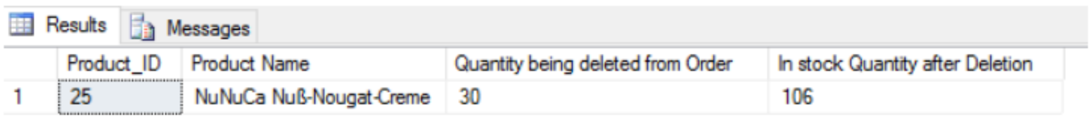
	product_name	unit_price	quantity_in_stock	supplier_name
1	Jack's New England Clam Chowder	10.1325	85	Silver Spring Wholesale Market
2	Jack's New England Clam Chowder	10.1325	85	Silver Spring Wholesale Market
3	Jack's New England Clam Chowder	10.1325	85	Silver Spring Wholesale Market
4	Jack's New England Clam Chowder	10.1325	85	Silver Spring Wholesale Market

Query compl... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (57) | master | 00:00:00 | 4 rows

4. Create a **DELETE** trigger on the order_details table to display the information shown below when you issue the following statement:

```
DELETE order_details
WHERE order_id=10001 AND product_id=25
```

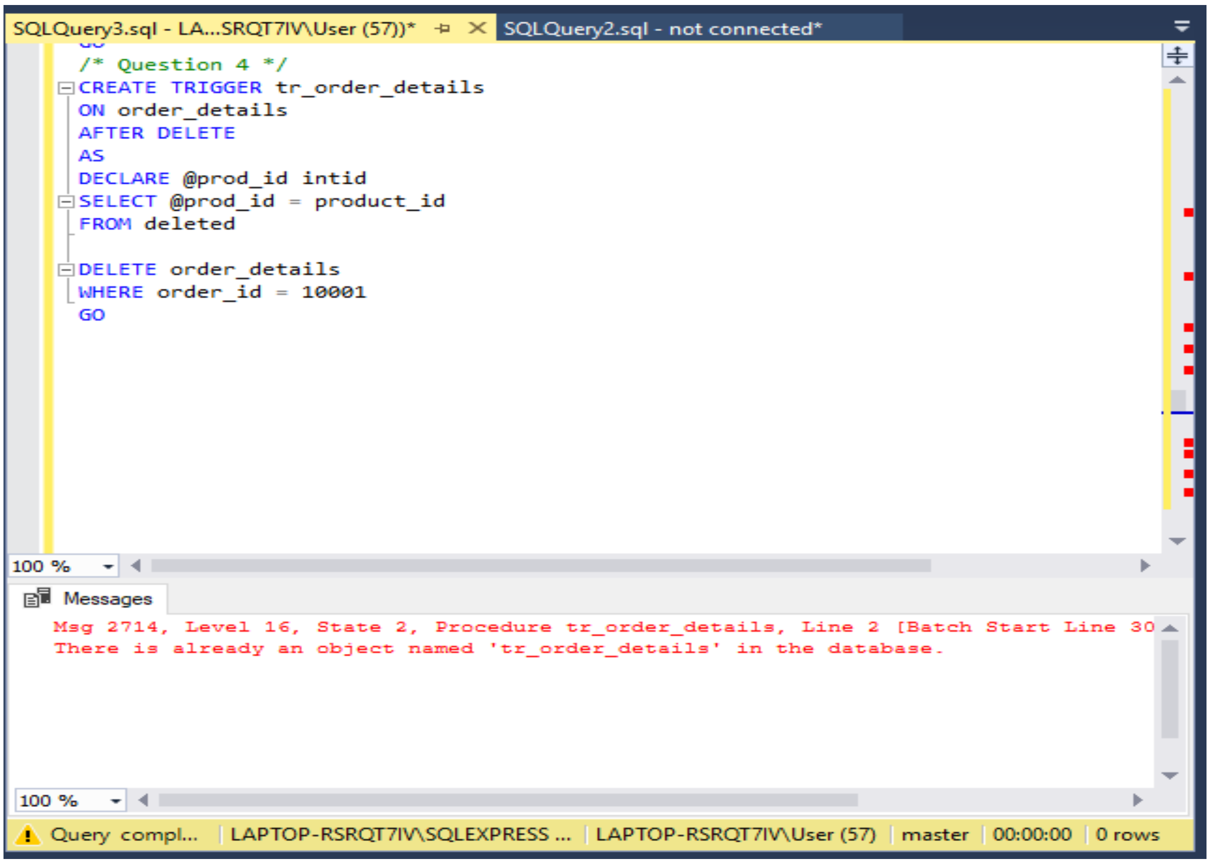
```
  You should get the following results:
```



	Product_ID	Product Name	Quantity being deleted from Order	In stock Quantity after Deletion
1	25	NuNuCa Nuß-Nougat-Creme	30	106

```
/* Question 4 */
CREATE TRIGGER tr_order_details
ON order_details
AFTER DELETE
AS
DECLARE @prod_id intid
SELECT @prod_id = product_id
FROM deleted
```

```
GO
DELETE order_details
WHERE order_id = 10001
GO
```



SQLQuery3.sql - LA...SRQT7IV\User (57))* | SQLQuery2.sql - not connected*

```
/* Question 4 */
CREATE TRIGGER tr_order_details
ON order_details
AFTER DELETE
AS
DECLARE @prod_id intid
SELECT @prod_id = product_id
FROM deleted

DELETE order_details
WHERE order_id = 10001
GO
```

Messages

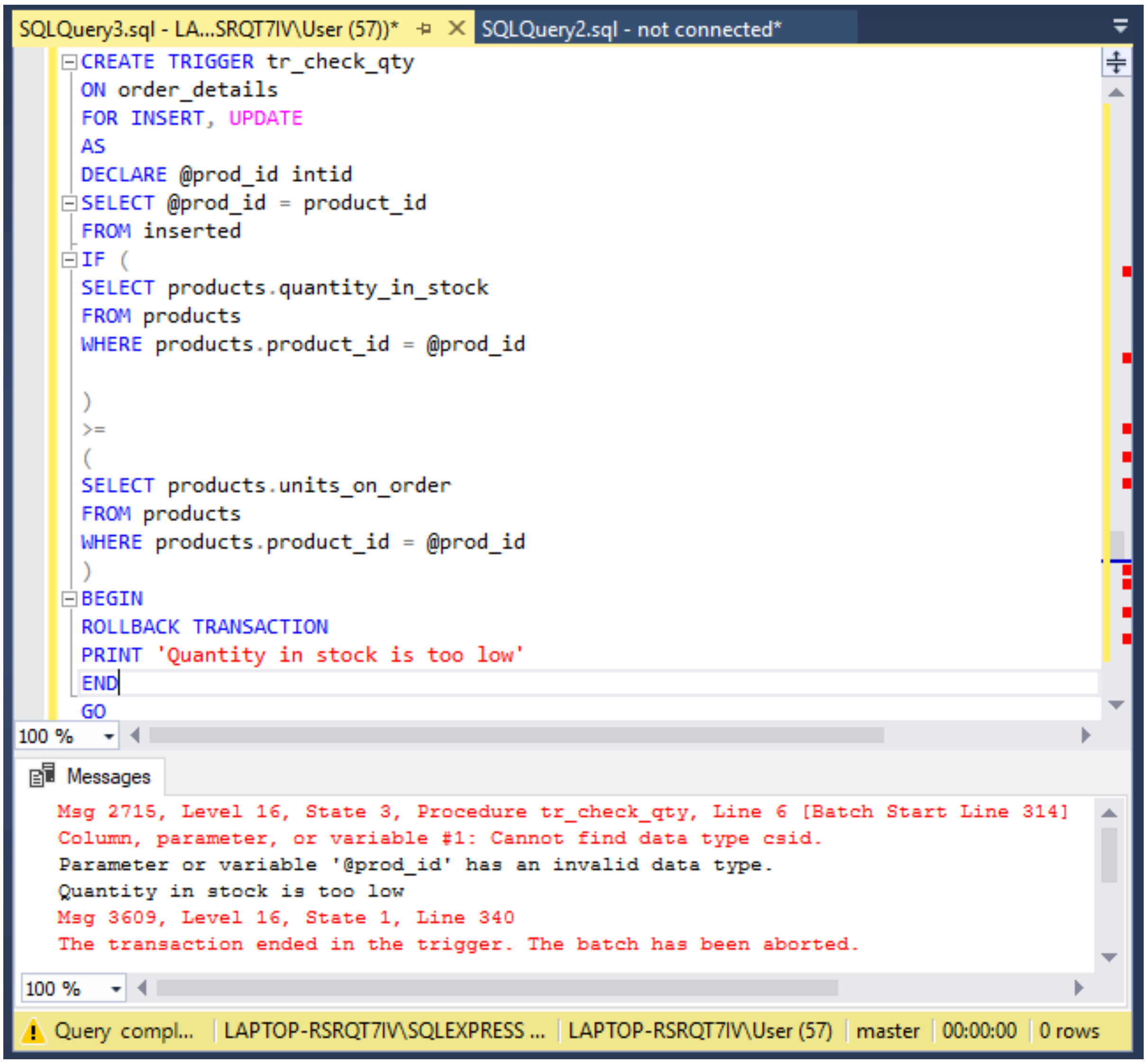
```
Msg 2714, Level 16, State 2, Procedure tr_order_details, Line 2 [Batch Start Line 30
There is already an object named 'tr_order_details' in the database.
```

Query compl... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (57) | master | 00:00:00 | 0 rows

5. Create an **INSERT** and **UPDATE** trigger called **tr_check_qty** on the order_details table to only allow orders of products that have a quantity in stock greater than or equal to the units ordered. Run the following query to verify your trigger.

   UPDATE order_details
   SET quantity = 30
   WHERE order_id = '10044'
       AND product_id = 7

```
/* Question 5 */
CREATE TRIGGER tr_check_qty
ON order_details
FOR INSERT, UPDATE
AS
DECLARE @prod_id intid
SELECT @prod_id = product_id
FROM inserted
IF (
SELECT products.quantity_in_stock
FROM products
WHERE products.product_id = @prod_id

)
>=
(
SELECT products.units_on_order
FROM products
WHERE products.product_id = @prod_id
)
BEGIN
ROLLBACK TRANSACTION
PRINT 'Quantity in stock is too low'
END
GO
UPDATE order_details
SET quantity = 30
WHERE order_id = '10044' AND product_id = 7
GO
```



```
CREATE TRIGGER tr_check_qty
ON order_details
FOR INSERT, UPDATE
AS
DECLARE @prod_id intid
SELECT @prod_id = product_id
FROM inserted
IF (
SELECT products.quantity_in_stock
FROM products
WHERE products.product_id = @prod_id

)
>=
(
SELECT products.units_on_order
FROM products
WHERE products.product_id = @prod_id
)
BEGIN
ROLLBACK TRANSACTION
PRINT 'Quantity in stock is too low'
END
GO
```

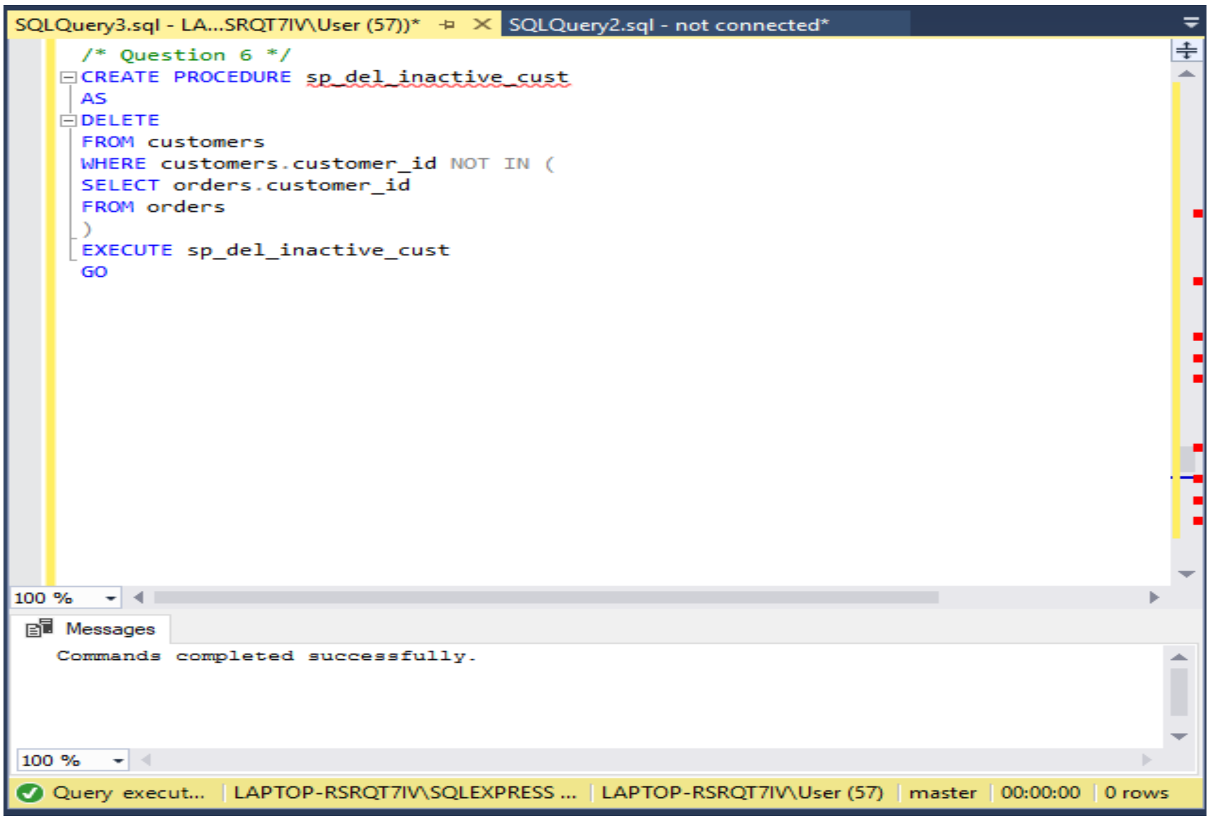
```
Msg 2715, Level 16, State 3, Procedure tr_check_qty, Line 6 [Batch Start Line 314]
Column, parameter, or variable #1: Cannot find data type csid.
Parameter or variable '@prod_id' has an invalid data type.
Quantity in stock is too low
Msg 3609, Level 16, State 1, Line 340
The transaction ended in the trigger. The batch has been aborted.
```

SQLQuery3.sql - LA...SRQT7IV\User (57))*
SQLQuery2.sql - not connected*
100 %
Messages
100 %
Query compl... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (57) | master | 00:00:00 | 0 rows

6. Create a stored procedure called **sp_del_inactive_cust** to **delete** customers that have no orders. The stored procedure should delete **1** row.

```
/* Question 6 */
CREATE PROCEDURE sp_del_inactive_cust
AS
DELETE
FROM customers
```

```
WHERE customers.customer_id NOT IN (
SELECT orders.customer_id
FROM orders
)
EXECUTE sp_del_inactive_cust
GO
```


SQLQuery3.sql - LA...SRQT7IV\User (57))*
SQLQuery2.sql - not connected*
100 %
Messages

```
/* Question 6 */
CREATE PROCEDURE sp_del_inactive_cust
AS
DELETE
FROM customers
WHERE customers.customer_id NOT IN (
SELECT orders.customer_id
FROM orders
)
EXECUTE sp_del_inactive_cust
GO
```

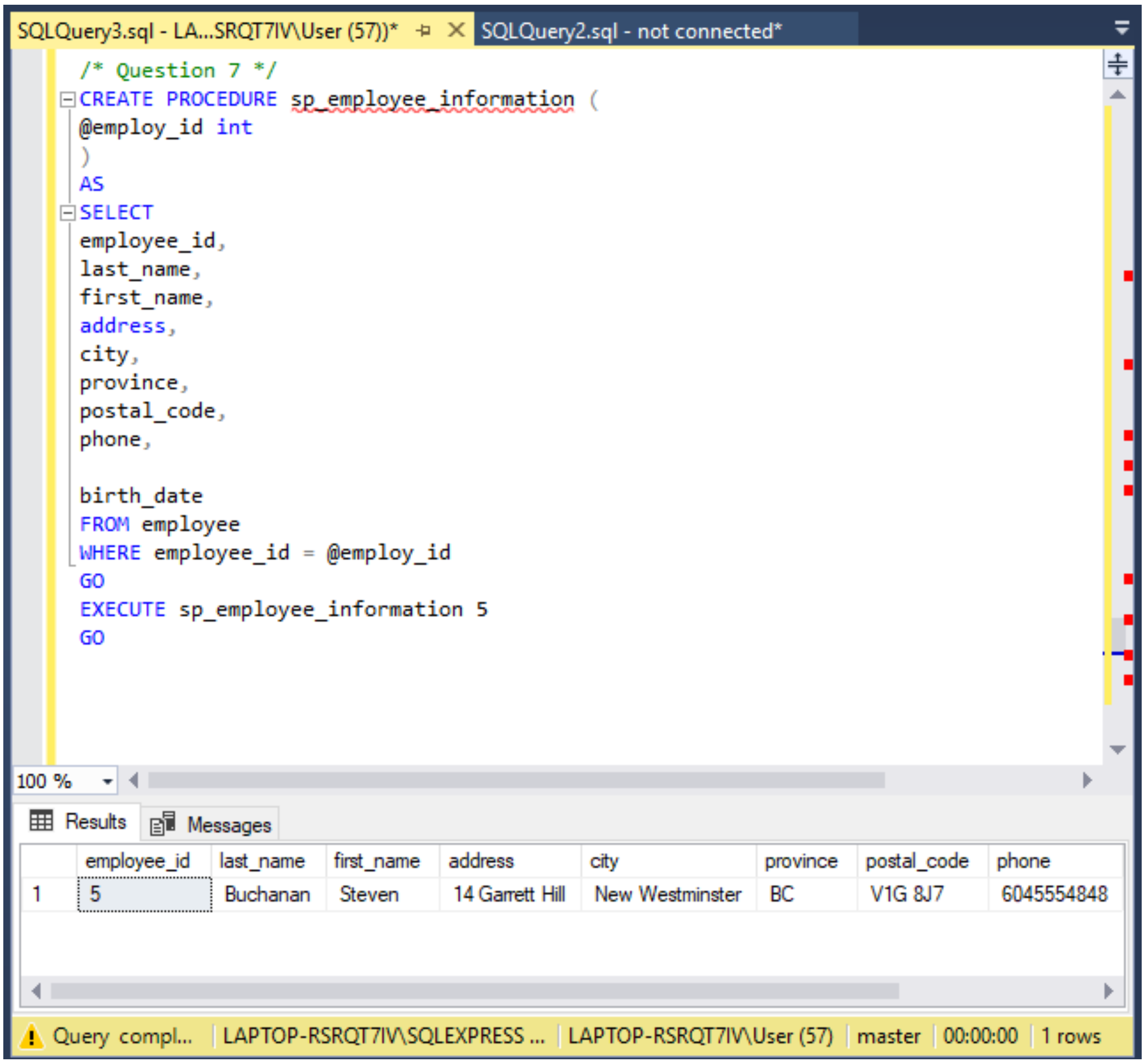
```
Commands completed successfully.
```

```
Query execut... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (57) | master | 00:00:00 | 0 rows
```

7. Create a stored procedure called **sp_employee_information** to display the employee information for a particular employee. The **employee id** will be an **input parameter** for the stored procedure. Run the stored procedure displaying information for employee id of **5**. The stored procedure should produce the result set listed below.

```
employee_id  last_name  first_name  address          city                 province postal_code phone        birth_date
-----------  ---------  ----------  ---------------  -------------------- -------- ----------- ------------ -----------------------
5            Buchanan   Steven      14 Garrett Hill  New Westminster BC            V1G 8J7     6045554848   1955-03-04 00:00:00.000

(1 row(s) affected)
```

```
/* Question 7 */
CREATE PROCEDURE sp_employee_information (
@employ_id int
)
AS
SELECT
employee_id,
last_name,
first_name,
address,
city,
province,
postal_code,
phone,
birth_date
FROM employee
WHERE employee_id = @employ_id
GO
EXECUTE sp_employee_information 5
GO
```


SQLQuery3.sql - LA...SRQT7IV\User (57))*   SQLQuery2.sql - not connected*

```
/* Question 7 */
CREATE PROCEDURE sp_employee_information (
@employ_id int
)
AS
SELECT
employee_id,
last_name,
first_name,
address,
city,
province,
postal_code,
phone,

birth_date
FROM employee
WHERE employee_id = @employ_id
GO
EXECUTE sp_employee_information 5
GO
```

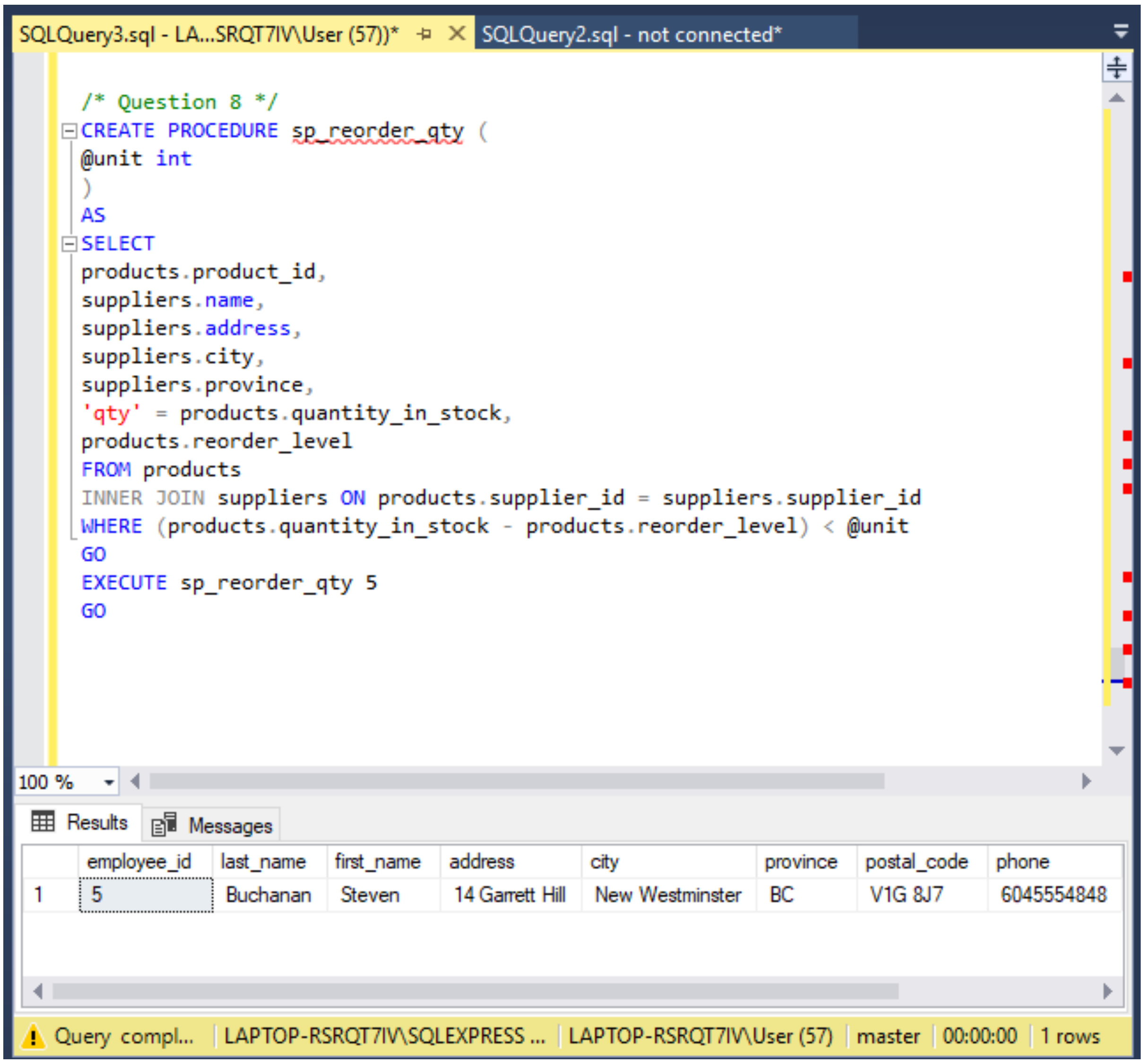
	employee_id	last_name	first_name	address	city	province	postal_code	phone
1	5	Buchanan	Steven	14 Garrett Hill	New Westminster	BC	V1G 8J7	6045554848

Query compl... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (57) | master | 00:00:00 | 1 rows

8. Create a stored procedure called **sp_reorder_qty** to show when the reorder level subtracted from the quantity in stock is less than a specified value. The **unit** value will be an **input parameter** for the stored procedure. Display the product id, quantity in stock, and reorder level from the products table, and the supplier name, address, city, and province from the suppliers table. Run the stored procedure displaying the information for a value of **5**. The stored procedure should produce the result set listed below.

```
product_id    name                             address                city              province qty reorder_level
------------- -------------------------------- ---------------------- ----------------- -------- --- -------------
2             Edward's Products Ltd.           1125 Howe Street       Vancouver          BC      17  25
3             Edward's Products Ltd.           1125 Howe Street       Vancouver          BC      13  25
5             New Orlean's Spices Ltd.         1040 Georgia Street    West Vancouver     BC      0   0
11            Armstrong Company                1638 Derwent Way       Richmond           BC      22  30
17            Steveston Export Company         2951 Moncton Street    Richmond           BC      0   0
...
68            Dare Manufacturer Ltd.           1603 3rd Avenue        West  Burnaby      BC      6   15
70            Steveston Export Company         2951 Moncton Street    Richmond           BC      15  30
```

```sql
/* Question 8 */
CREATE PROCEDURE sp_reorder_qty (
@unit int
)
AS
SELECT
products.product_id,
suppliers.name,
suppliers.address,
suppliers.city,
suppliers.province,
'qty' = products.quantity_in_stock,
products.reorder_level
FROM products
INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
WHERE (products.quantity_in_stock - products.reorder_level) < @unit
GO
EXECUTE sp_reorder_qty 5
GO
```


SQLQuery3.sql - LA...SRQT7IV\User (57))*
SQLQuery2.sql - not connected*
100 %
Results
Messages

```
/* Question 8 */
CREATE PROCEDURE sp_reorder_qty (
@unit int
)
AS
SELECT
products.product_id,
suppliers.name,
suppliers.address,
suppliers.city,
suppliers.province,
'qty' = products.quantity_in_stock,
products.reorder_level
FROM products
INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
WHERE (products.quantity_in_stock - products.reorder_level) < @unit
GO
EXECUTE sp_reorder_qty 5
GO
```

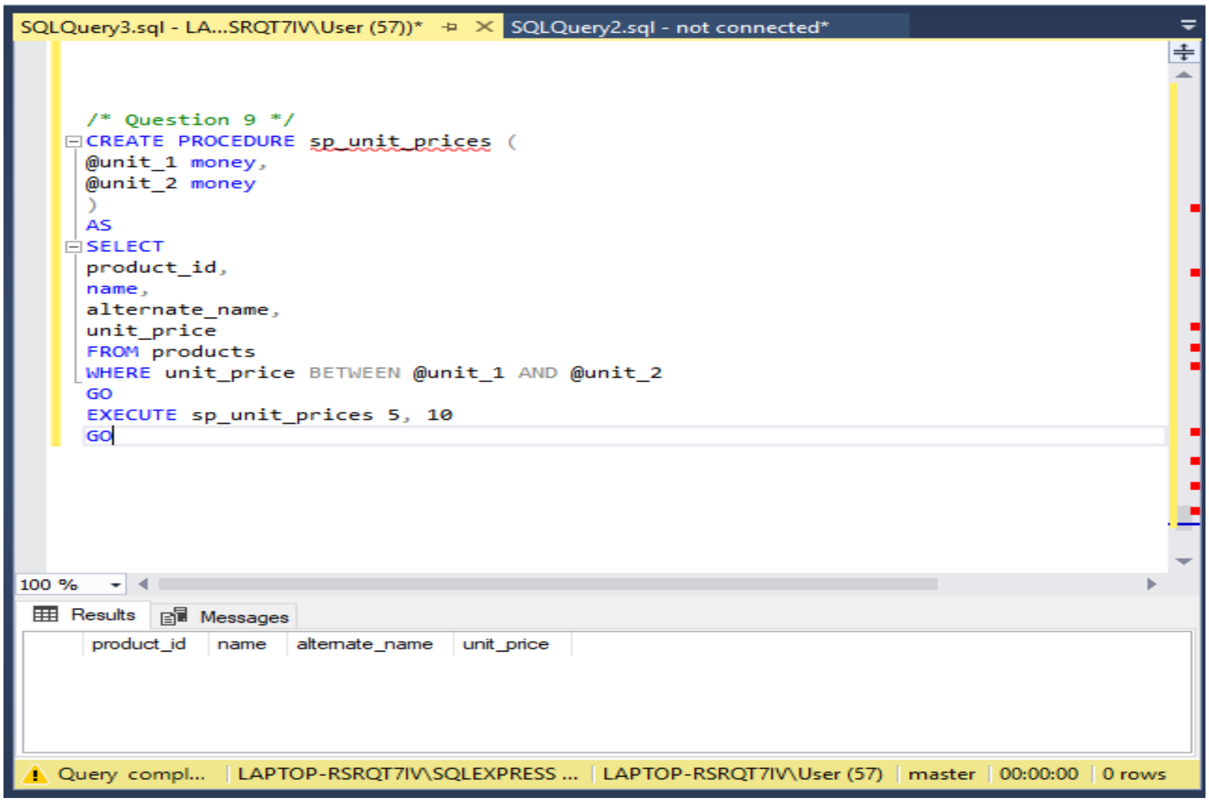
	employee_id	last_name	first_name	address	city	province	postal_code	phone
1	5	Buchanan	Steven	14 Garrett Hill	New Westminster	BC	V1G 8J7	6045554848

Query compl... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (57) | master | 00:00:00 | 1 rows

9. Create a stored procedure called **sp_unit_prices** for the product table where the **unit price** is **between particular values**. The **two unit prices** will be **input parameters** for the stored procedure. Display the product id, product name, alternate name, and unit price from the products table. Run the stored procedure to display products where the unit price is between **$5.00** and **$10.00**. The stored procedure should produce the result set listed below.

```
product_id    name                              alternate_name                    unit_price
-------------  --------------------------------  --------------------------------  -------------
13            Konbu                             Kelp Seaweed                      6.30
19            Teatime Chocolate Biscuits        Teatime Chocolate Biscuits        9.66
23            Tunnbr÷d                          Thin Bread                        9.45
45            R°gede sild                       Smoked Herring                    9.975
47            Zaanse koeken                     Zaanse Cookies                    9.975
52            Filo Mix                          Mix for Greek Filo Dough          7.35
54            TourtiÞre                         Pork Pie                          7.8225
75            Rh÷nbrõu Klosterbier              Rh÷nbrõu Beer                     8.1375

(8    row(s) affected)
```

```
/* Question 9 */
CREATE PROCEDURE sp_unit_prices (
@unit_1 money,
@unit_2 money
)
AS
SELECT
product_id,
name,
alternate_name,
unit_price
FROM products
WHERE unit_price BETWEEN @unit_1 AND @unit_2
GO
EXECUTE sp_unit_prices 5, 10
GO
```



SQLQuery3.sql - LA...SRQT7IV\User (57))* SQLQuery2.sql - not connected*

```
/* Question 9 */
CREATE PROCEDURE sp_unit_prices (
@unit_1 money,
@unit_2 money
)
AS
SELECT
product_id,
name,
alternate_name,
unit_price
FROM products
WHERE unit_price BETWEEN @unit_1 AND @unit_2
GO
EXECUTE sp_unit_prices 5, 10
GO
```

product_id	name	alternate_name	unit_price

Query compl... | LAPTOP-RSRQT7IV\SQLEXPRESS ... | LAPTOP-RSRQT7IV\User (57) | master | 00:00:00 | 0 rows

3. SUMMARY

This intensive project gave hands-on experience on how to use tools and processes for data modelling in Relational Database Management System and also focus on Structured Query Language to define and manipulate data. All the questions have been completed with evidence of their result.

4. CHALLENGES

Since I have MacBook, so I had to download Docker and Azure Data Studio in order to start my project and run SQL statements but then it had some issues so I couldn’t run it, so I had to switch the computer, in the project I got stuck with question 4 and question 5 in part D and I couldn’t figure it out how to resolve it.

5. SCRIPT

```
/* -----------------------part B---------------------- */
/* question 1 */
SELECT customer_id, name, city, country
FROM customers
ORDER BY customer_id;
GO
/* Question 2 */

ALTER TABLE customers
ADD active BIT NOT NULL
CONSTRAINT default_active DEFAULT(1);
GO

/* Question 3 */
SELECT
orders.order_id,
'product_name' = products.name,
'customer_name' = customers.name,
'order_date' = CONVERT(char(11), orders.order_date, 100),
'new_shipped_date' = CONVERT(char(11), orders.shipped_date + 7,100),
'order_cost' = (order_details.quantity * products.unit_price)
FROM orders
INNER JOIN order_details ON orders.order_id = order_details.order_id
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN customers ON customers.customer_id = orders.customer_id
WHERE orders.order_date BETWEEN 'Jan 1 2001' AND 'Dec 31 2001'
GO
/* Question 4 */
SELECT

orders.customer_id,
'name' = customers.name,
customers.phone,
orders.order_id,
orders.order_date
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
WHERE shipped_date IS NULL
ORDER BY name
GO
/* Question 5 */
SELECT
customers.customer_id,
customers.name,
customers.city,
titles.description
FROM customers
```

```sql
INNER JOIN titles ON customers.title_id = titles.title_id
WHERE customers.region IS NULL
GO

/* Question 6 */
SELECT
'supplier_name' = suppliers.name,
'products_name' = products.name,
products.reorder_level,
products.quantity_in_stock
FROM suppliers
INNER JOIN products ON suppliers.supplier_id = products.supplier_id
WHERE products.reorder_level > products.quantity_in_stock
ORDER BY supplier_name
GO

/* Question 7 */
SELECT
orders.order_id,
customers.name,
customers.contact_name,
'shipped_date' = CONVERT(char(11), orders.shipped_date, 100),
'elapsed' = DATEDIFF(YEAR, orders.shipped_date, 'Jan 1 2008')
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
WHERE orders.shipped_date IS NOT NULL
GO

/* Question 8 */
SELECT
'name' = SUBSTRING(name, 1,1),
'total' = COUNT(name)
FROM customers
GROUP BY SUBSTRING(name, 1, 1)
HAVING COUNT(name) >= 2 AND SUBSTRING(name, 1,1) != 'S'
GO

/* Question 9 */
SELECT
order_details.order_id,
order_details.quantity,
products.product_id,
products.reorder_level,
suppliers.supplier_id
FROM order_details
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
WHERE order_details.quantity > 100
ORDER BY order_details.order_id
GO

/* Question 10 */
SELECT
product_id,
name,
quantity_per_unit,
unit_price
FROM products
WHERE name LIKE '%tofu%' OR name LIKE '%chef%'
ORDER BY name
GO

/* ------------------------ Part C --------------------------- */
/* Question 1 */
CREATE TABLE employee (
employee_id int NOT NULL,
last_name varchar(30) NOT NULL,
first_name varchar(15) NOT NULL,

address varchar(30),
city varchar(20),
province char(2),
postal_code varchar(7),
phone varchar(10),
birth_date datetime NOT NULL
);
GO
/* Question 2 */
ALTER TABLE employee
ADD PRIMARY KEY (employee_id)
GO
/* Question 3 */
BULK INSERT employee
FROM 'C:\TextFiles\employee.txt'
WITH (
CODEPAGE=1252,
DATAFILETYPE = 'char',
FIELDTERMINATOR = '\t',
KEEPNULLS,
ROWTERMINATOR = '\n'
)
ALTER TABLE orders
ADD CONSTRAINT fk_employee_orders FOREIGN KEY (employee_id)
REFERENCES employee(employee_id);
GO
/* Question 4 */
INSERT INTO shippers(name)
VALUES('Quick Express')
GO
/* Question 5 */
UPDATE products
SET unit_price = unit_price * 1.05
WHERE unit_price >= 5 AND unit_price <= 10
GO
/* Question 6 */
UPDATE customers

SET fax = 'Unknown'
WHERE fax IS NULL
GO
/* Question 7 */
CREATE VIEW vw_order_cost
AS
SELECT
orders.order_id,
orders.order_date,
products.product_id,
customers.name,
'order_cost' = (order_details.quantity * products.unit_price)
FROM orders
INNER JOIN order_details ON order_details.order_id = orders.order_id
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN customers ON orders.customer_id = customers.customer_id
GO
SELECT * FROM vw_order_cost
WHERE order_id BETWEEN 10000 AND 10200
```

```
GO
/* Question 8 */
CREATE VIEW vw_list_employees
AS
SELECT * FROM employee
GO
SELECT
employee_id,
'name' = last_name + ', ' + first_name,
'birth_date' = convert(char(10), birth_date, 102)
FROM vw_list_employees
WHERE employee_id = 5 OR employee_id = 7 OR employee_id = 9
GO
/* Question 9 */
CREATE VIEW vw_all_orders
AS
SELECT
orders.order_id,
orders.shipped_date,
customers.customer_id,
'customer_name' = customers.name,

customers.city,
customers.country
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
GO
SELECT
order_id,
customer_id,
customer_name,
city,
country,
'shipped_date' = CONVERT(char(11), shipped_date, 100)
FROM vw_all_orders
WHERE shipped_date BETWEEN 'Jan 1 2002' AND 'Dec 31 2002'
ORDER BY customer_name, country
GO
/* Question 10 */
CREATE VIEW vw_supplier_products_shipped
AS
SELECT
suppliers.supplier_id,
'supplier_name' = suppliers.name,
products.product_id,
'product_name' = products.name
FROM suppliers
INNER JOIN products ON products.supplier_id = suppliers.supplier_id
GO
SELECT * FROM vw_supplier_products_shipped
GO
/* ------------------------ Part D --------------------------- */
/* Question 1 */
CREATE PROCEDURE sp_customer_city (
@city varchar(30)
)
AS
SELECT
customer_id,
name,
address,

city,
phone
FROM customers
WHERE city = @city
GO
EXECUTE sp_customer_city 'London'
GO




/* Question 2 */
CREATE PROCEDURE sp_orders_by_dates (
@start datetime,
@end datetime
)
AS
SELECT
orders.order_id,
orders.customer_id,
'customer_name' = customers.name,
'shipper_name' = shippers.name,
orders.shipped_date
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
INNER JOIN shippers ON orders.shipper_id = shippers.shipper_id
WHERE shipped_date BETWEEN @start AND @end
GO
EXECUTE sp_orders_by_dates 'Jan 1 2003', 'Jun 30 2003'
GO
/* Question 3 */
CREATE PROCEDURE sp_product_listing (
@product varchar(50),
@month varchar(8),
@year int
)
AS
SELECT
'product_name' = products.name,
products.unit_price,
products.quantity_in_stock,
'supplier_name' = suppliers.name
FROM products
INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
INNER JOIN order_details ON products.product_id =
order_details.product_id
INNER JOIN orders ON order_details.order_id = orders.order_id

WHERE products.name LIKE '%' + @product + '%'
AND DATENAME(Month, orders.order_date) = @month
AND DATENAME(Year, orders.order_date) = @year
GO
EXECUTE sp_product_listing 'Jack', June, 2001
GO
/* Question 4 */
CREATE TRIGGER tr_order_details
ON order_details
AFTER DELETE
AS
DECLARE @prod_id intid
SELECT @prod_id = product_id
FROM deleted


```

```
DELETE order_details
WHERE order_id = 10001
GO
```

```

/* Question 5 */
CREATE TRIGGER tr_check_qty
ON order_details
FOR INSERT, UPDATE
AS
DECLARE @prod_id intid
SELECT @prod_id = product_id
FROM inserted
IF (
SELECT products.quantity_in_stock
FROM products
WHERE products.product_id = @prod_id

)
>=
(
SELECT products.units_on_order
FROM products
WHERE products.product_id = @prod_id
)
BEGIN
ROLLBACK TRANSACTION
PRINT 'Quantity in stock is too low'
END
GO
UPDATE order_details
SET quantity = 30
WHERE order_id = '10044' AND product_id = 7
GO
/* Question 6 */
CREATE PROCEDURE sp_del_inactive_cust
AS
DELETE
FROM customers
WHERE customers.customer_id NOT IN (
SELECT orders.customer_id
FROM orders
)
EXECUTE sp_del_inactive_cust
GO

/* Question 7 */
CREATE PROCEDURE sp_employee_information (
@employ_id int
)
AS
SELECT
employee_id,
last_name,
first_name,
address,
city,
province,
postal_code,
phone,

birth_date
FROM employee
WHERE employee_id = @employ_id
GO
```

```sql
EXECUTE sp_employee_information 5
GO

/* Question 8 */
CREATE PROCEDURE sp_reorder_qty (
@unit int
)
AS
SELECT
products.product_id,
suppliers.name,
suppliers.address,
suppliers.city,
suppliers.province,
'qty' = products.quantity_in_stock,
products.reorder_level
FROM products
INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
WHERE (products.quantity_in_stock - products.reorder_level) < @unit
GO
EXECUTE sp_reorder_qty 5
GO


/* Question 9 */
CREATE PROCEDURE sp_unit_prices (
@unit_1 money,
@unit_2 money
)
AS
SELECT
product_id,
name,
alternate_name,
unit_price
FROM products
WHERE unit_price BETWEEN @unit_1 AND @unit_2
GO
EXECUTE sp_unit_prices 5, 10
 GO
```